Escuela Politécnica Superior

UNIVERSIDAD CARLOS III DE MADRID

ESTUDIO DE SOLUCIÓN BASADA EN SISTEMA NOSQL COMO SUSTITUCIÓN DEL SISTEMA DE DIRECTORIO USADO EN EL DEPARTAMENTO DE INFORMÁTICA

TRABAJO FINAL DE GRADO

Titulación:

Grado en Ingeniería Informática

Trabajo realizado por:

Sergio Aragonés Tercero

Trabajo dirigido por:

Alejandro Calderón Mateos

3 DE FEBRERO DE 2017

Índice de contenidos

Índice de ilustraciones

Índice de tablas

Agradecimientos

Puede que éste sea el apartado más complicado de todos, simplemente por la dificultad que conlleva para mí plasmar en un documento escrito todos los sentimientos que me inundan al pensar lo que significa terminar este proyecto. Porque éste no es un trabajo más, es el culmen de 5 años de un esfuerzo titánico que, al final, te das cuenta que he merecido mucho más que la pena. Han sido 5 años de risas y llantos, buenos y malos momentos, alivio y sufrimiento, alegría y desesperación; 5 años de importantes decisiones y grandes equivocaciones, 5 años conociéndome más a mí mismo, y a aquellos que me han acompañado.

Cuando queda tan poco para terminar un camino tan largo, no puedes evitar echar la vista atrás y agradecer infinitamente a la gente que te ha ayudado a llegar hasta ahí. Porque sabes que sin su ayuda te habrías quedado a mitad de camino. Sin sus ánimos no hubieras sido capaz de levantarte todas aquellas veces en las que caías rendido al suelo, y dándote a veces el empujoncito necesario para superar esos obstáculos que, a priori, parecían tan insalvables. Yo todo eso lo sé, y por eso quiero agradecérselo dedicándoles unas palabras en el trabajo que significa el fin de una etapa en mi vida.

En primer lugar, a mi familia, por permitirme estar aquí hoy. Gracias al esfuerzo y sacrificio de mis padres, yo he podido tener todo lo que ellos no tuvieron en su día, como una educación que me permita labrarme un futuro mejor que el que ellos han tenido. Gracias también por la educación personal que me habéis inculcado, por los valores que ahora he tomado como míos y que, en un futuro, espero transmitir a mis hijos. Gracias a mi madre, esa incansable luchadora, aliada y enemiga en muchas ocasiones, pero que, a pesar de las disputas, no puedo dejar de querer por todo lo que ha hecho por mí desde que nací. Gracias por no darte nunca por vencida conmigo, ni siquiera cuando aún no había nacido. Gracias por tus lecciones de vida, por enseñarme que con esfuerzo y constancia todo es posible, porque en la vida nadie da duros a pesetas. Gracias también a ese ser intrascendente que ha velado por mi seguridad incluso antes de mi nacimiento, por hacerme parte de ese mínimo porcentaje de éxito. Y a mis abuelos, por cuidarme, educarme, protegerme y quererme, gracias por hacerme llegar hasta aquí.

A mis compañeros de prácticas, las Pringles Paprika y Google. Por todos esos viernes hasta las once de la noche, algunos con resultados infructuosos, y otros con momentos imborrables. Ha merecido la pena el esfuerzo. A mis amigos universitarios, los nuevos y los viejos, los Turbados y los Pingponeros. Gracias por hacerme más llevadera la estancia estos años y hacerme vivir una de las mejores etapas.

Gracias también a los profesores, que con su esfuerzo y afán de enseñanza no se dieron por vencidos en el intento de ayudar a muchos de nosotros a seguir adelante. En especial a mi tutor Alejandro Calderón, ya que sin sus consejos e indicaciones no habría podido terminar este trabajo.

Por último, a todos los que estén empezando, o a los que les quede poco para terminar, recordad siempre que todo esfuerzo tiene su recompensa.

S.A.T.

Abstract

NoSQL term is not about databases that do not use SQL as query language, is about databases that do not use SQL as the only query language. These databases can use other types of query languages, like JSON or GQL. The main feature of these NoSQL databases, as opposed to relational databases, is that they do not use a relational model to store data. That is the difference between them.

The main issue here is that there is a huge amount of NoSQL databases, and all of them are mostly alike. So, how can I know what NoSQL database I have to choose? This document includes a manual that can answer that question. In order to do that, a path is offered to the user with the aim to help him to decide what NoSQL database is best for the data he is managing. Moreover, after studying all these NoSQL systems, I will be able to propose one to the Computer Science and Engineering Department, in order to replace the current LDAP.

1 – Introducción

En la actualidad, las tecnologías NoSQL están en un continuo crecimiento debido a las mejoras que presentan respecto a los sistemas SQL tradicionales. Cada vez más empresas se decantan por un sistema NoSQL como medio para almacenar la información, debido a la facilidad que tienen éstas para manejar grandes cantidades de datos y a la rapidez con la que permiten acceder a ellos.

Sin embargo, existe una gran cantidad de SGBD NoSQL, y pocas o ninguna guía de recomendación sobre cuál usar en cada situación. Por ello, una de las finalidades de este Trabajo de Fin de Grado es servir de guía sobre qué sistema NoSQL usar, dependiendo de la situación que tenga cada usuario. La otra finalidad de este proyecto es, utilizando esta guía sobre bases de datos NoSQL, ser capaz de escoger una y proponerla como sustitución del sistema LDAP, usado actualmente para almacenar los datos del Departamento de Informática de la Universidad Carlos III de Madrid.

En dicho trabajo, primero se hará un estudio sobre el término NoSQL y qué sistemas se basan en él en la actualidad, de donde se sacarán cuatro grupos fundamentales, poniéndose de ejemplo dos bases de datos de cada grupo. A continuación, se realizará una comparativa según las características que garantizan los sistemas NoSQL, de donde saldrá el árbol de decisión que usarán los usuarios para elegir el sistema más acorde a sus necesidades. Después, utilizando dicho árbol de decisión, se escogerá uno de los sistemas puestos de ejemplo anteriormente como sistema alternativo al LDAP. Y, por último, se describirán las características que debe tener el sistema elegido, así como el proceso de implantación y prueba con los datos sacados del LDAP.

1.1 – Motivación

La principal motivación que me ha llevado a realizar este proyecto ha sido la escasez de conocimientos sobre bases de datos que poseo, ya que no ha sido uno de mis puntos fuertes a lo largo de la carrera. Por ello, cuando los responsables del Laboratorio de Informática de la Universidad me comunicaron que iban a prescindir del sistema de LDAP que usaban actualmente, les propuse pasar los datos a un sistema NoSQL.

Sin embargo, como he comentado anteriormente, la cantidad de gestores NoSQL es altísima, por lo que antes de migrar los datos necesitaba saber a qué gestor lo iba a hacer. De este modo, decidí añadir un estudio a mi proyecto en el que comparara varios SGBD NoSQL y decidiera cuál de ellos sería más apropiado para la tarea que me proponía.

1.2 – Objetivos

La finalidad de este proyecto es proponer un sistema de almacenamiento alternativo al LDAP usado actualmente para guardar los datos del Departamento de Informática. Por tanto, es necesario estudiar a fondo la tecnología NoSQL con el fin de poder reducir las posibles alternativas, y luego compararlas entre sí para ver cuál de ellas es más apropiada como sustituta del LDAP. Para ello se han marcado los siguientes objetivos, que deberán verse cumplidos a la finalización del proyecto:

- Estudiar las posibles alternativas que pueden valer como sustitución del sistema LDAP, y compararlas con la solución propuesta.
- Estudiar y analizar la diversidad de sistemas de gestión NoSQL que existen en el mercado actualmente.
- Ofrecer al usuario una guía de elección de sistemas NoSQL.
- Elegir, utilizando la guía de elección, uno de los sistemas estudiados como sustituto del LDAP.

Una vez que se elija el sistema, éste debe cumplir una serie de objetivos:

- El sistema debe ser capaz de realizar las mismas tareas que el LDAP, como son añadir, modificar, borrar y buscar usuarios.
- El sistema debe ser capaz de realizar otras tareas, como la de insertar usuarios con distintos atributos, ya sea de uno en uno o desde un archivo.
- El sistema debe poder implantarse en los equipos de los técnicos del Laboratorio (para este proyecto sólo hace falta Linux).
- El sistema debe permitir sacar los datos a un archivo siempre que se quiera.
- El sistema debe permitir la gestión de los usuarios que pueden acceder a él.
- El sistema debe permitir la gestión del acceso de forma remota.

Además, se debe buscar una solución que no signifique un sobrecoste para la Universidad, por lo que también deberá cumplir lo siguiente:

- El sistema deberá ser de licencia gratuita y código abierto.
- El mantenimiento del sistema deberá ser poco costoso en horas por persona y en coste de equipos.

1.3 – Resto del documento

En este punto se indica cómo está estructurado el documento. El proyecto se divide en 11 apartados, los cuales se especifican a continuación:

- **1 – Introducción.** En este apartado se hace una breve introducción sobre el tema a tratar en el proyecto y qué pasos se van a seguir.
- **2 – Estado de la cuestión.** En este apartado se explica en qué estado se encuentra el tema a tratar, las bases de datos NoSQL en este caso, las posibles soluciones que se podrían haber tomado en lugar de la que se propone en este documento, y, por último, una comparativa entre varios tipos de NoSQL para guiar al usuario en la elección de una base de datos NoSQL.
- **3 – Análisis.** En este apartado se indica el ciclo de vida del proyecto, el marco regulador del proyecto, la solución que se ha elegido y qué características tiene, los requisitos que debe cumplir dicha solución, así como una matriz para verificar el origen de los requisitos, y, por último, la descripción del entorno donde se ejecutará el sistema elegido.
- **4 – Diseño.** En este apartado se define la arquitectura que tendrá el sistema y el diseño de la base de datos, se especifican los diagramas de casos de uso, de clases y de secuencia, y, por último, se describe la interfaz del sistema.
- **5 – Implementación**. En este apartado se explica el funcionamiento del código desarrollado.
- **6 – Implantación.** En este apartado se explica el proceso de obtención de los datos que serán usados en las pruebas, y el proceso de implantación del sistema elegido.
- **7 – Evaluación.** En este apartado se describe el funcionamiento y el resultado de las pruebas a las que es sometido el sistema, con el fin de comprobar que cumple con las necesidades establecidas por el cliente.
- **8 – Planificación y presupuesto.** En este apartado se definen las fases del proyecto y su duración, así como el precio desglosado del mismo y su impacto socioeconómico.
- **9 – Conclusiones y trabajos futuros.** En este apartado se exponen las conclusiones finales del proyecto y los trabajos futuros que pueden salir del mismo.
- **Anexo.** En este apartado se encuentra un apéndice con los acrónimos utilizados durante el proyecto y su explicación, otro con un resumen de 10 páginas del proyecto en inglés, otro apéndice con la guía de instalación de las herramientas utilizadas, otro con la descripción de los programas creados en la implementación y, por último, un apéndice con ejemplos prácticos de la solución elegida.
- **Bibliografía**. En este apartado se indican las fuentes bibliográficas de las que se ha extraído la información consultada durante la realización del proyecto.

2 – Estado de la cuestión

Este apartado se va a dividir en 3 partes. En la primera se describe cómo se encuentra actualmente el estado de la cuestión que se estudia en este trabajo, es decir, las bases de datos NoSQL; en la segunda se estudian las posibles soluciones que se podrían haber tomado para sustituir el LDAP; y en la tercera y última se hará una comparativa entre varios tipos distintos de SGBD, atendiendo a sus características.

2.1 – Introducción

El término NoSQL se originó en 1998 para referirse a una base de datos relacional y de código abierto que no usaba SQL como lenguaje de consultas, pero no fue hasta 2009 cuando se asoció este término a las bases de datos no relacionales que estaban surgiendo en esa época. Por tanto, Johan Oskarsson organizó una reunión para aglutinar todos estos nuevos sistemas bajo una misma denominación [1].

Hasta entonces, el SQL era el lenguaje más usado por los motores de bases de datos de las empresas, que podían ser escaladas si necesitaban más capacidad de cómputo por el aumento de la información. Sin embargo, con la llegada de la web y los servicios en la nube, las bases de datos SQL no son capaces de dar servicio a una gran cantidad de usuarios consultando la misma información al mismo tiempo (Google, Amazon, Facebook). "*Con todo ello llegaron los problemas de alta escalabilidad* [2]". Aunque, como he mencionado anteriormente, las bases de datos podrían ser escaladas, conllevaba un gran esfuerzo hacerlo, a veces con consultas de triples y cuádruples JOINs poco eficientes.

Por tanto, aunque el NoSQL pierda la capacidad de realizar JOINs o realizar operaciones entre diferentes colecciones de datos, "*los sistemas NoSQL intentan atacar este problema proponiendo una estructura de almacenamiento más versátil* [2]", lo que, unido a su escalabilidad, sobre todo horizontal al ser capaces de poner más nodos a realizar la misma tarea para disminuir el tiempo, lo hacen más apto para consultas entre grandes volúmenes de datos.

El término NoSQL no hace referencia a que no se use el lenguaje SQL, si no a que no es el único que se usa. Otros motores de bases de datos usan JSON o GQL. Lo que diferencia básicamente a las bases de datos SQL de las NoSQL es que las últimas no usan modelos relacionales, por tanto, como Carlo Strozzi, el inventor del término NoSQL en 1998 dijo, a estas bases de datos se las debería llamar NoRel. Sin embargo, el término NoSQL está tan extendido que, a sabiendas de que es un error, seguiremos diciendo bases de datos NoSQL, para no confundir al lector.

Una vez introducido el término NoSQL, y sabiendo ya qué características nos aportan estos sistemas, hay que decir que existen una enorme cantidad de ellos divididos en múltiples grupos según su funcionalidad. Algunos de ellos, como los XML, Orientados A Objetos, Grid & Cloud, Multimodel o Multidimensional, entre otros [3], no los estudiaremos en este trabajo. Sin embargo, en las webs [4], [5] y [6], como en la

mayoría, proponen una clasificación basada sólo en los siguientes tipos, clasificación sobre la que realizaremos el estudio:

- **Documentales**. Almacenan la información de forma semiestructurada, es decir, en documentos cuyo formato puede ser JSON o BSON. Estos documentos son similares a los registros de una BBDD relacional, con la diferencia de que no son tan rígidos; es decir, un documento puede tener unos atributos distintos a los de otro documento, y ambos estar en la misma BBDD. Los documentos están direccionados por una clave única mediante la cual se recupera la información, pero estas BBDD también cuentan con un API que permite recuperar la información asociada a un campo o campos específicos. Según [6], "*son las bases de datos NoSQL más versátiles, ya que se pueden utilizar en gran cantidad de proyectos, incluyendo muchos que tradicionalmente funcionarían sobre bases de datos relacionales*". La gran ventaja de estos sistemas es que puedes introducir un documento entero directamente sin tener que mapearlo primero a ningún modelo, como el entidad-relación, siempre que lo pases a formato JSON primero. Son sistemas que bien pueden ser consistentes (que los datos se actualizan inmediatamente después de ser modificados) o eventualmente consistentes (que los datos se actualizan pasado un cierto período de tiempo después de ser modificados). Ejemplos: CouchDB y MongoDB.
- **Columnares**. Almacenan la información en forma de columnas. Cuentan con una clave primaria para obtener y actualizar los datos. Son parecidas a las BBDD relacionales, pero utilizando columnas en lugar de registros. Están orientadas al tratamiento de grandes cantidades de datos. Según [5], "*las bases de datos clave-valor y orientadas a columna ofrecen un modelo de consultas limitado que puede imponer costes de desarrollo y requisitos a nivel de aplicación para ofrecer un modelo de consultas avanzado. Un ejemplo de esto son los índices, que deben ser gestionados por el propio usuario*". Son sistemas eventualmente consistentes. Ejemplos: Cassandra y HBase.
- **Clave-valor**. Almacenan la información mediante tuplas que contienen una clave y un valor. Al igual que las columnares, también cuentan con una clave primaria para obtener y actualizar los datos. A la hora de obtener información, simplemente hay que buscar por la clave para recuperar el valor. Como cuentan en [4] "*este grupo de bases de datos NOSQL, cuyo precursor fue BigTable de Google, tiene un modelo con pares clave-valor especialmente útiles para problemas de escrituras masivas de streaming*". Son sistemas eventualmente consistentes. Ejemplo: DynamoDB y Redis.
- **Grafo**. Almacenan la información en forma de grafos, utilizando los nodos (se muestran mediante círculos) y aristas (se muestran mediante flechas) para representar los datos almacenados. Los nodos representan las entidades, y pueden tener cualquier número de atributos, cada uno como un par clave-valor, así como una etiqueta indicando su rol en el dominio del grafo. Las aristas representan las relaciones entre los nodos, siempre tienen dirección, tipo, nodo inicial y nodo final, y, al igual que éstos, pueden tener cualquier número de atributos [7]. Aunque tengan dirección, las relaciones pueden ser recorridas en ambos sentidos. Gracias a que estas relaciones se almacenan correctamente, dos nodos pueden compartir cualquier número o tipo de

relaciones sin que ello suponga una bajada de rendimiento. Son especialmente útiles para modelos con muchas relaciones, ya que son más eficientes para consultas donde haya relaciones de proximidad entre los datos que para consultas globales. Pueden ser consistentes o eventualmente consistentes. Ejemplo: Neo4j y InfiniteGraph.

En la siguiente imagen se explica gráficamente cómo almacenan la información cada uno de estos tipos [8].

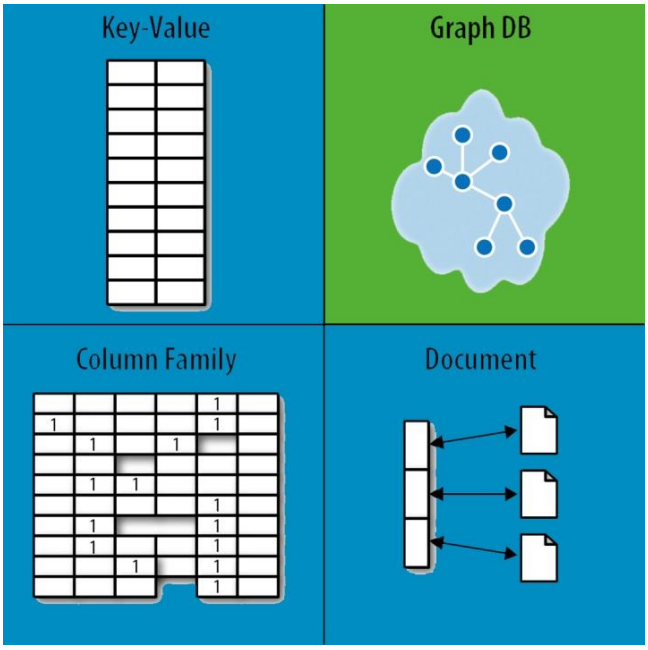

Ilustración 1: Cómo almacenan la información las BBDD NoSQL

El problema radica en que, ni siquiera reduciendo el número a sólo 4 tipos de bases de datos distintas podemos estar seguros de cuál de ellos va mejor para lo que podamos necesitar. Para ello nos haría falta una guía que pudiéramos seguir, con el objetivo de identificar qué sistema va a ser más efectivo con los datos que tengamos y las operaciones que necesitemos hacer.

Por tanto, en la sección *2.3 – COMPARATIVA* realizaré una comparativa entre varias bases de datos, con el fin de orientar sobre cuál de ellas va mejor en cada situación.

2.2 – Estudio de posibles soluciones

Aunque una base de datos NoSQL es una buena opción para almacenar los datos contenidos actualmente en el LDAP, no es la única opción que se podría aplicar. Hay dos opciones más que he barajado como posibles soluciones pero que, por diversos motivos, he decidido rechazar en pro de una base de datos NoSQL. A continuación, expondré dichas soluciones y los motivos que me han llevado a decantarlas.

2.2.1 – Base de datos relacional

Las bases de datos relacionales (BDR) son aquellas que cumplen con el modelo relacional, que es el más extendido a la hora de implementar bases de datos. La interfaz estándar de una BDR es el Lenguaje de Consultas Estructuradas (SQL por sus siglas en inglés, *Structured Query Language*), que es utilizado mediante comandos para acceder a la información almacenada en las BDR. Estas bases de datos pueden considerarse como un conjunto de tablas, cada una de las cuales tiene distintas categorías predefinidas en forma de columnas, e instancias de datos para dichas categorías en forma de filas. A su vez, las diferentes tablas pueden relacionarse entre sí para navegar por ellas recuperando la información [9].

A continuación, vamos a ver cuáles serían las ventajas e inconvenientes de usar una BDR como alternativa al LDAP.

Ventajas

- Garantiza que no habrá dos usuarios con los mismos datos.
- Garantiza que, si eliminamos a un usuario del sistema, toda la información relacionada con él en otras tablas también desaparecerá.
- Garantiza la integridad de los datos.
- Ofrecen sistemas sencillos para visualizar y manipular la información.

Inconvenientes

- Hay que mapear la información sacada del LDAP al Modelo Relacional.
- Si se añaden campos a la información de un usuario habría que revisar, y puede que rehacer, el Modelo Entidad-Relación.
- Muchas búsquedas al mismo tiempo por parte de distintos usuarios provocarían muchas JOINs, que ralentizarían el sistema.
- No trabajan de forma sencilla con bloques de texto como tipo de datos.

2.2.2 – Contactos de Google

Contactos de Google es una herramienta gratuita de administración de contactos de Google, que viene disponible con cualquier cuenta de Gmail. Es una agenda online que puedes sincronizar en todos tus dispositivos y acceder a ella en cualquier parte. En ella puedes guardar la información asociada a distintos usuarios, como el nombre, una foto o el correo electrónico, entre otros.

A continuación, vamos a ver cuáles serían las ventajas e inconvenientes de usar los Contactos de Google como alternativa al LDAP.

Ventajas

- Puedes agrupar los contactos por grupos, como por Departamento o Categoría, por ejemplo.
- Cuenta con una barra de búsqueda que hace más sencillo encontrar a los usuarios.
- Puedes importar los contactos directamente desde un archivo.

Inconvenientes

- Para compartir la información almacenada en la agenda debes exportar primero los contactos a un archivo, que luego importarán los usuarios a su cuenta de Gmail.
- Sólo puedes almacenar 25.000 contactos, de 128 Kb cada uno, y no superando entre todos los 20 MB [10].
- Dispone de unos campos predefinidos y no da la opción de añadir unos nuevos, por lo que información como el Área de Conocimiento habría que introducirlo en otro campo como el del Cumpleaños, o no introducirlo, perdiendo así esta información.
- Aunque existe una herramienta para buscar contactos duplicados, dichos contactos debes borrarlos tú, no se borran automáticamente o te impide introducirlos para que no tengas que eliminarlos más adelante.

2.2.3 – Archivo de Excel en Google Drive

Microsoft Excel es una aplicación desarrollada por Microsoft y que se caracteriza por ser un software de hojas de cálculo. Es una aplicación dedicada a fórmulas matemáticas y lógicas, aunque puede utilizarse para guardar cualquier tipo de información en sus celdas, pues acepta tanto valores numéricos como alfanuméricos. Excel permite que se modifiquen los bordes de las celdas, de modo que se pueden formar tablas con ellas. Por su parte, Google Drive es un servicio de alojamiento gratuito para almacenar archivos, disponible para cualquier usuario con una cuenta de Gmail. Dichos archivos pueden ser compartidos entre múltiples usuarios, siendo el creador el encargado de conceder permisos de lectura y/o escritura al archivo.

A continuación, vamos a ver cuáles serían las ventajas e inconvenientes de usar Excel como alternativa al LDAP.

Ventajas

- Para obtener el archivo con los datos sólo es necesario acceder a la URL en la que se encuentra alojado el mismo, sin tener que iniciar sesión en una cuenta de Gmail.
- Los datos son visualmente agradables a la vista de cualquier usuario, pues puedes organizarlos en forma de tablas con colores.

Inconvenientes

- Posee muchas restricciones, como puede verse en [11], que no pueden ser solucionadas mediante escalamiento, por ejemplo.
- Cada vez que haya que introducir o eliminar información de un usuario, hay que buscarlo previamente.

2.2.4 – Resumen

Para terminar, en la siguiente tabla se muestran qué características cumplen las soluciones anteriores y cuáles no. En ella también se puede apreciar cómo la solución que se propone cumple con todas y cada una de ellas, siendo ésta la razón principal de que se haya escogido una base de datos NoSQL como alternativa final al LDAP.

Características/Solución	BBDD Relacional	Contactos de Google	Excel en Google Drive	BBDD NoSQL
No almacenar datos duplicados	✓	✘	✘	✓
Facilidad de añadir más campos	✘	✓ 2	✓ 4	✓
Búsquedas masivas	✘	✓ 3	✓ 3	✓
Facilidad de los usuarios para acceder a la información	✓	✓	✓	✓
Escalabilidad	✓ 1	✘	✘	✓ 1

Tabla 1: Comparativa entre soluciones

NOTAS:

1: En la BDR sería vertical, y hasta cierto punto, y en la BBDD NoSQL sería horizontal.

2: Hasta el límite de campos que existen.

3: Depende más del servidor donde esté el archivo que de la herramienta en sí.

4: Hasta el límite de columnas.

2.3 – Comparativa

Para realizar esta comparativa, voy a partir del Teorema CAP. Dicho teorema también es llamado Teorema de Brewer en honor a la conjetura que surgió en el año 2000 de boca de Eric Brewer, de la Universidad de Berkeley, y que posteriormente fue probada por Seth Gilbert y Nancy Lynch, del MIT, pasando de conjetura a teorema. Lo que postula este teorema es que cualquier sistema distribuido no puede garantizar al mismo tiempo las siguientes características [12]:

- ***Consistency* (Consistencia)**: se garantiza la consistencia cuando se realiza una modificación en un nodo y todos los nodos restantes ven el mismo resultado.
- ***Availability* (Disponibilidad)**: se garantiza la disponibilidad cuando cualquier petición a un nodo del sistema recibe una respuesta.
- ***Partition tolerance* (Tolerancia a particiones)**: se garantiza la tolerancia a particiones cuando, a pesar de que algún nodo se separe de la red, el sistema sigue procesando las peticiones.

Por tanto, el Teorema CAP sólo nos garantiza que se cumplan dos de esas tres características:

- **CA**: el sistema garantiza que todos los nodos estén disponibles y que los datos sean consistentes, pero no garantiza que se traten las peticiones si un nodo se separa de la red.
- **AP**: el sistema garantiza que todas las peticiones serán tratadas, aunque los nodos se separen de la red, pero no garantiza que los datos sean consistentes.
- **CP**: el sistema garantiza que los datos serán consistentes y que las peticiones serán tratadas, aunque los nodos se separen de la red, pero no garantiza que el sistema responda siempre.

Sin embargo, para solucionar este problema, muchos sistemas recurren al modelo de consistencia BASE (***B**asic **A**vailability, **S**oft State, **E**ventual Consistency*), que garantiza las 3 cualidades anteriores, aunque de forma muy básica cada una de ellas. De esta forma, el sistema puede estar disponible la mayor parte del tiempo, aunque no el 100% del mismo, en un estado de consistencia débil, para que los datos sean consistentes en un futuro, como en una lectura, o sean siempre consistentes, pero sólo en ciertas instantáneas procesadas con anterioridad [8].

En la siguiente imagen tenemos un ejemplo de Teorema CAP con algunas BBDD y qué características cumple cada una [13]. La palabra CAP viene de las iniciales en inglés de las 3 características: ***C**onsistency*, ***A**vailability*, ***P**artition tolerance*.

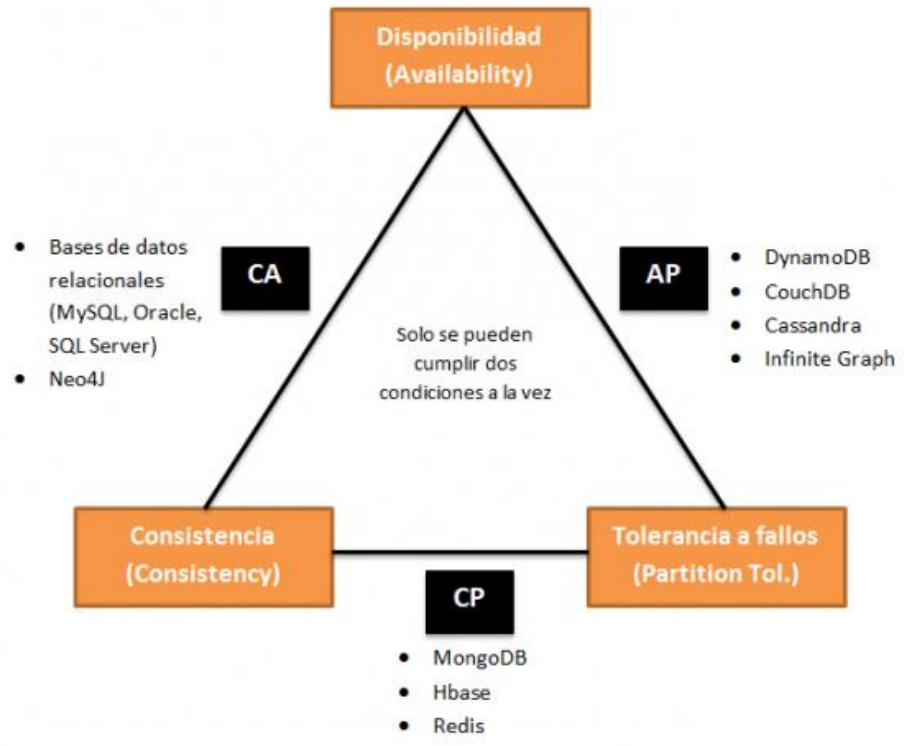

Ilustración 2: Teorema CAP

En este punto, el usuario que lea esta guía para averiguar qué base de datos usar debe decidir qué características quiere que presente su sistema, y decantarse por dos de ellas, puesto que hemos visto que no hay ningún sistema que garantice las tres al mismo tiempo. Dependiendo de qué características priorice el usuario, dispondrá de unas bases de datos distintas, como se aprecia en la imagen anterior.

2.3.1 – Consistencia y Disponibilidad (CA)

Según [12], este es el caso más improbable que pueda darse en un sistema, ya que para ello es necesario que la comunicación entre los nodos esté siempre en perfecto funcionamiento y sin fallos. Si el sistema no está particionado, podemos garantizar consistencia y disponibilidad; pero si el sistema se particiona, como no tenemos tolerancia a particiones, debería fallar, pero entonces no estaríamos garantizando la disponibilidad, y nos encontraríamos en el mismo caso que un sistema CP.

Si aun así el usuario quiere un sistema CA, las opciones que baraja son una base de datos relacional o la única opción NoSQL que garantiza consistencia y disponibilidad: **Neo4j**. Empresas que la utilizan: eBay, Cisco o Hewlett-Packard [14].

2.3.2 – Disponibilidad y Tolerancia a particiones (AP)

En este caso nos importa más que el sistema esté siempre disponible ante cualquier situación, aunque los datos a veces no sean consistentes entre sí. Para este tipo de sistema tenemos las siguientes opciones:

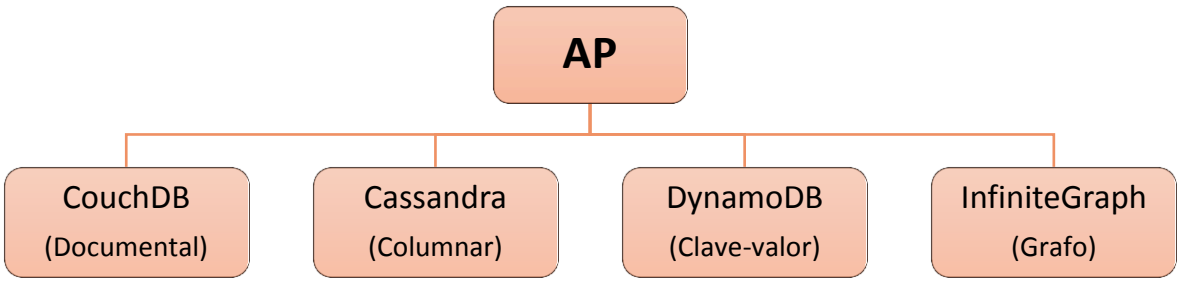

Ilustración 3: Esquema de elección de BBDD tipo AP

Ahora es decisión del usuario elegir cuál de ellos va a usar. La decisión va a depender de cómo tenga almacenada la información actualmente:

- **CouchDB:** Esta es la mejor elección si tiene la información almacenada en archivos con un esquema de tipo arbóreo, es decir, que los campos pueden tener subcampos, y éstos a su vez subcampos, como si fueran las ramas de un árbol, ya que al ser de tipo documental puedes guardar esta información en formato JSON. Empresas que la utilizan: BBC o Credit Suisse [15].

- **Cassandra:** Esta es la mejor elección si tiene la información almacenada de forma que un parámetro o atributo tiene muchos valores, pues ésta es la manera que tienen las bases de datos columnares de almacenar la información. Como explicaron en [16], Cassandra es una de las mejores opciones si se tienen muchos datos y se quieren consultar rápidamente. Empresas que la utilizan: Facebook, Twitter o Digg [17].
- **DynamoDB:** Esta es la mejor elección si tiene la información almacenada de forma que cada parámetro o atributo tiene un solo valor, pues ésta es la manera que tienen las bases de datos clave-valor de almacenar la información. Empresas que la utilizan: Amazon.
- **InfiniteGraph:** Esta es la mejor elección si tiene la información almacenada de modo que los datos estén relacionados entre sí, como en un modelo relacional, ya que las bases de datos tipo grafo guardan la información como entidades unidas entre sí por líneas. Empresas que la utilizan: Boeing, Raytheon o General Dynamics [18].

2.3.3 – Consistencia y Tolerancia a particiones (CP)

En este caso, para garantizar la consistencia, tenemos que asegurarnos de que las escrituras se realizan en todos los nodos, para que los datos sean los mismos. Si el sistema no está particionado, la petición es procesada por todos los nodos y se mantiene la consistencia; pero si el sistema se particiona, unos nodos pueden no recibir la petición, por lo que los que sí la reciben, para garantizar la consistencia, deben rechazar la operación hasta que se deshaga la partición, dando lugar a indisponibilidad. Para este tipo de sistema tenemos las siguientes opciones:

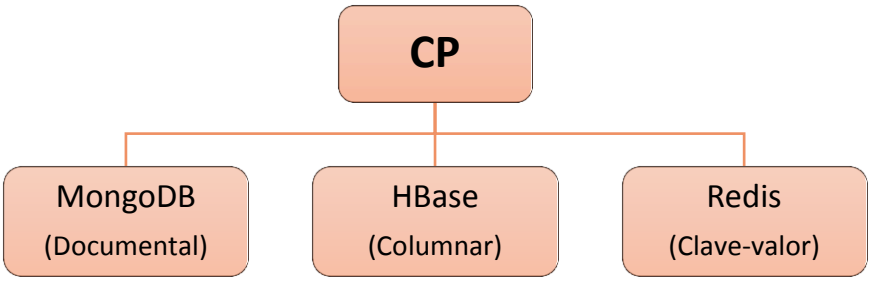

Ilustración 4: Esquema de elección de BBDD de tipo CP

Ahora es decisión del usuario elegir cuál de ellos va a usar. La decisión va a depender de cómo tenga almacenada la información actualmente:

- **MongoDB:** Esta es la mejor elección si quieres guardar documentos directamente. Lo único que tendrías que hacer es pasar los datos a formato JSON o CSV, y luego importar el documento entero en MongoDB. Como explicaron en [16], MongoDB es una de las mejores opciones para agregar datos. Empresas que la utilizan: Electronic Arts, Barclays o Telefónica [19].

- **HBase:** Esta es la mejor elección si tiene la información almacenada de forma que un parámetro o atributo tiene muchos valores, pues ésta es la manera que tienen las bases de datos columnares de almacenar la información. Como explicaron en [16], HBase es una de las mejores opciones si se tienen muchos datos y se quieren consultar rápidamente. Empresas que la utilizan: Adobe, Meetup o Yahoo [20].
- **Redis:** Esta es la mejor elección si tiene la información almacenada de forma que cada parámetro o atributo tiene un solo valor, pues ésta es la manera que tienen las bases de datos clave-valor de almacenar la información. Empresas que la utilizan: GitHub, Snapchat o Flickr [21].

2.3.4 – Resumen

En resumen, el usuario que lea este documento deberá elegir qué dos características de las descritas en *2.3 – COMPARATIVA* desea que presente su sistema y de cuál de ellas va a prescindir, pues no se pueden garantizar las tres a la vez.

Una vez que haya elegido dichas cualidades, deberá elegir entre las bases de datos que se le ofrecen en función de cómo tenga almacenados los datos en ese momento. La siguiente imagen describe, a modo de esquema, el proceso de selección del sistema final, partiendo primero de la elección de las dos características, y posteriormente de la elección de la base de datos.

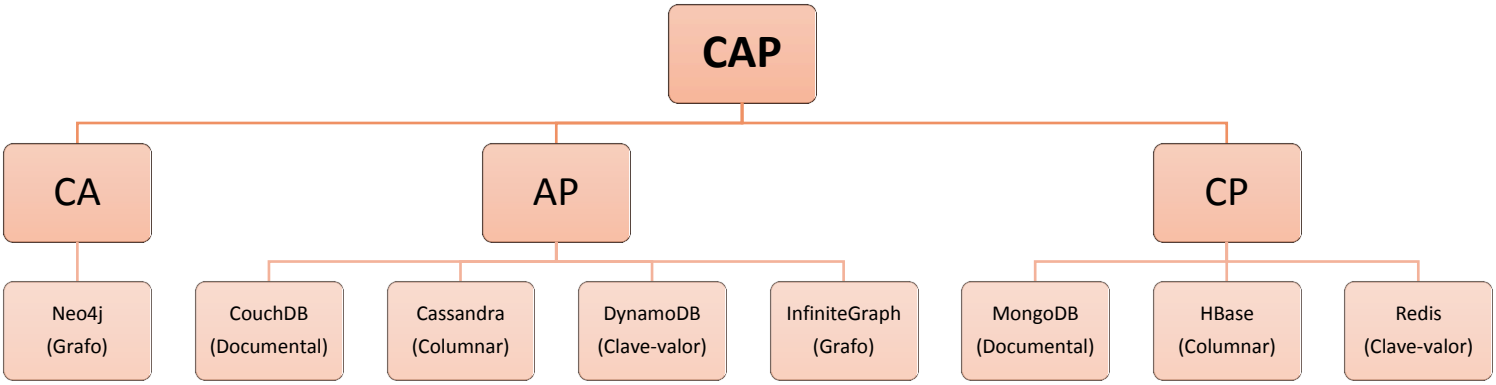

Ilustración 5: Esquema de elección de BBDD según CAP

Sin embargo, si el usuario desconoce qué características necesita su sistema o no tiene predilección por ninguna de ellas en especial, y únicamente sabe cómo tiene almacenada actualmente la información, se le ofrece la posibilidad de escoger qué tipo de base de datos quiere, y después elegir entre las opciones que se le dan en función de las características que garantiza cada sistema. La siguiente imagen describe, a modo de esquema, un proceso de selección inverso al descrito anteriormente, partiendo primero de la elección del tipo de base de datos que quiere, de entre los cuatro tipos descritos en *2.1 – INTRODUCCIÓN*, y luego de la base de datos, en función de las cualidades que garantiza cada una de ellas.

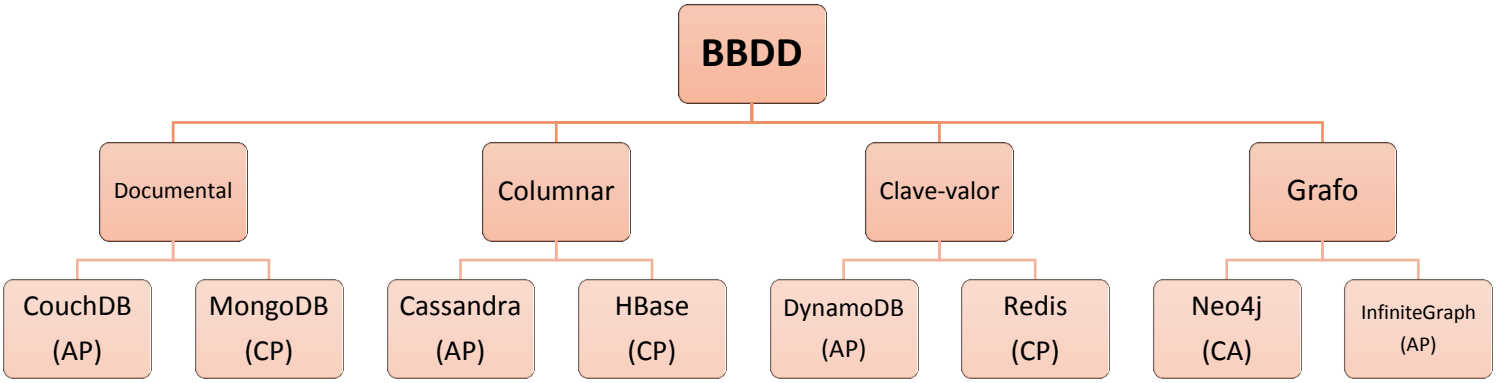

Ilustración 6: Esquema de elección de BBDD según tipo

3 – Análisis

Este apartado se va a dividir en 4 partes. En la primera se describe el ciclo de vida del proyecto y el marco regulador del mismo; en la segunda se indican las características de la solución deseada y cuál se ha elegido finalmente; en la tercera se definen los requisitos que el sistema debe cumplir para satisfacer las necesidades del cliente, y se hace una revisión final de los requisitos utilizando una matriz de trazabilidad; y en la cuarta y última se indica el entorno operacional donde funcionará el sistema.

La metodología que se ha usado para este documento está basada en el estándar para proyectos de software que define la ESA [22] y Métrica v3 [23], la cual está inspirada en los estándares internacionales ISO/IEC 12207 [24] e ISO/IEC 15504 [25]. Este documento utiliza dicha metodología sólo como estructura general, porque el uso de Métrica v3 está más orientado a la planificación, desarrollo y mantenimiento de software.

3.1 – Ciclo de vida

El sistema que voy a proponer como alternativa al LDAP tiene un ciclo de vida circular y constante, que se mantendrá hasta que se decida dejar de utilizarlo y pasar a otro sistema de almacenamiento distinto. Este ciclo de vida se basa en una serie de etapas que se irán cumpliendo en orden, tal y como indica la siguiente imagen. Ésta ha sido desarrollada por medio de la herramienta Draw.io [26].

Ilustración 7: Ciclo de vida del sistema

Las etapas del ciclo de vida son las siguientes:

1. **Análisis**. En esta etapa se recogen los requisitos que se deben cumplir para garantizar el funcionamiento del sistema.
2. **Diseño**. En esta etapa se define la forma en que el sistema cumplirá con los requisitos recogidos anteriormente.
3. **Implantación**. En esta etapa se instala el sistema y se documenta su forma de uso, de modo que sirva de alguna manera para entrenar a futuros usuarios consumidores del mismo.
4. **Pruebas**. En esta etapa se realizarán una serie de pruebas para encontrar los posibles fallos que pudiera tener el sistema y subsanarlos. También servirá para verificar que todos los requisitos definidos anteriormente se cumplen de alguna manera.
5. **Mantenimiento**. En esta etapa se monitorizan las prestaciones del sistema y se mantiene y actualiza el mismo, de modo que, cuando sea necesario, los nuevos requisitos que surjan irán incorporándose al sistema siguiendo este ciclo de vida.

3.2 – Marco regulador

En este apartado se explica la relación del proyecto con la normativa vigente en el momento de su realización, así como todo lo relacionado con la propiedad intelectual del mismo.

En cuanto a la normativa vigente, existen dos leyes aplicables al proyecto [27]:

- **Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal**. Lo que aplica esta ley al proyecto es que se debe garantizar y proteger los datos personales de los miembros del Departamento de Informática. Para ello, se cifrarán los datos del personal antes de introducirlos en la base de datos. Se utilizará criptografía simétrica, más concretamente el cifrador de bloque AES, y el administrador del sistema será el poseedor de la clave de descifrado.
- **Ley 9/2014, de 9 de mayo, General de Telecomunicaciones**. Lo que aplica esta ley al proyecto es que se deben cifrar las comunicaciones cuando se acceda de manera remota al servidor.

En cuanto a la propiedad intelectual, hay dos partes diferenciadas: la propiedad intelectual del proyecto y la de las herramientas utilizadas para su desarrollo.

Sobre la propiedad intelectual del proyecto, existe una ley aplicable al mismo:

- **Real Decreto Legislativo 1/1996, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia**. Lo que aplica esta ley al proyecto es que la propiedad intelectual del mismo corresponde a mi persona, Sergio Aragonés Tercero, por el mero hecho de su creación. Su autoría me corresponde por llevar mi nombre en la portada.

Sobre la propiedad intelectual de las herramientas utilizadas para el desarrollo del proyecto, hay que destacar que todas ellas se distribuyen bajo licencias de software libre y código abierto, aprobadas por:

- **GNU General Public License**. Cygwin y VirtualBox.
- **Apache License**. Cassandra.

Sin embargo, también se ha usado software no libre, como Windows 10 y Microsoft Office. Este software ha sido adquirido gracias a las claves proporcionadas por Microsoft DreamSpark, mediante el acuerdo entre Microsoft y la Universidad Carlos III de Madrid por el programa MSDN Academic Alliance. De este modo, se ha utilizado el SO Windows 10 para la realización del proyecto y las herramientas de Microsoft Office, como Excel y Word, para la documentación del mismo.

Por último, a la hora de entregar este proyecto para su calificación se entrega también una hoja de autorización firmada tanto por mí como por mi tutor, Alejandro Calderón, para permitir que la Universidad Carlos III de Madrid pueda publicar abiertamente el proyecto en el Repositorio Institucional de la Universidad, otorgándole al proyecto las condiciones de uso de la licencia Creative Commons.

3.3 – Características de la solución deseada

La solución que vamos a proponer como alternativa al LDAP tiene que satisfacer los requerimientos de la Universidad, de los técnicos del laboratorio y de los equipos del mismo para poder ser implantada. Del mismo modo, debe garantizar las siguientes características, algunas de las cuales ya se definieron en *1.2 – OBJETIVOS*.

- El sistema debe permitir insertar y actualizar usuarios, tanto de uno en uno a través de la interfaz como cargando un archivo a la base de datos.
- El sistema debe permitir borrar un usuario o todos los que haya en la base de datos.
- El sistema debe permitir la búsqueda de usuarios por cualquiera de sus atributos.
- La interfaz gráfica del sistema debe mostrar el resultado de las consultas realizadas por el usuario.
- El sistema debe permitir la exportación de la información desde la base de datos a un archivo.

- El sistema debe restringir con unas credenciales las peticiones de inserción, actualización y borrado.
- El sistema debe permitir dar de alta, de baja o modificar los permisos de acceso de un usuario.
- El sistema debe permitir las conexiones remotas.
- El sistema debe poder ser implantado en los equipos actuales del Laboratorio del Departamento de Informática.
- El sistema debe poder escalarse horizontalmente.
- El sistema debe ser de código abierto y licencia gratuita.
- El mantenimiento del sistema debe ser poco costoso en horas por persona.

3.4 – Elección de la solución

Para elegir la solución que se propondrá como alternativa al LDAP vamos a usar el árbol de decisión creado en *2.3.4 – RESUMEN*.

Si usamos el primer árbol, las características que vamos a primar son Disponibilidad (A) y Tolerancia a particiones (P). Esto es así porque la función del sistema va a estar más orientada a recibir consultas que a inserciones, actualizaciones o borrados, ya que introducir, modificar o eliminar usuarios se hará en contadas ocasiones, como al inicio de la implantación, o cuando se añada, modifique o elimine la información de un miembro del Personal del Departamento de Informática, cosa que no ocurrirá muy a menudo. El resto del tiempo el sistema recibirá muchas consultas para acceder a la información de los usuarios almacenados en él. Por tanto, debemos garantizar que el sistema esté siempre disponible para recibir estas consultas, aunque se produzca alguna partición. Además, algunos sistemas ofrecen consistencia eventual, es decir, que las escrituras sobre la base de datos no se realizarán de inmediato, sino al cabo de un rato, pero es algo que podemos poner en segundo plano para favorecer la disponibilidad.

En la siguiente imagen se muestra el árbol de decisión tipo CAP, con el camino elegido marcado en azul. De este camino salen 4 opciones: CouchDB, Cassandra, DynamoDB e InfiniteGraph.

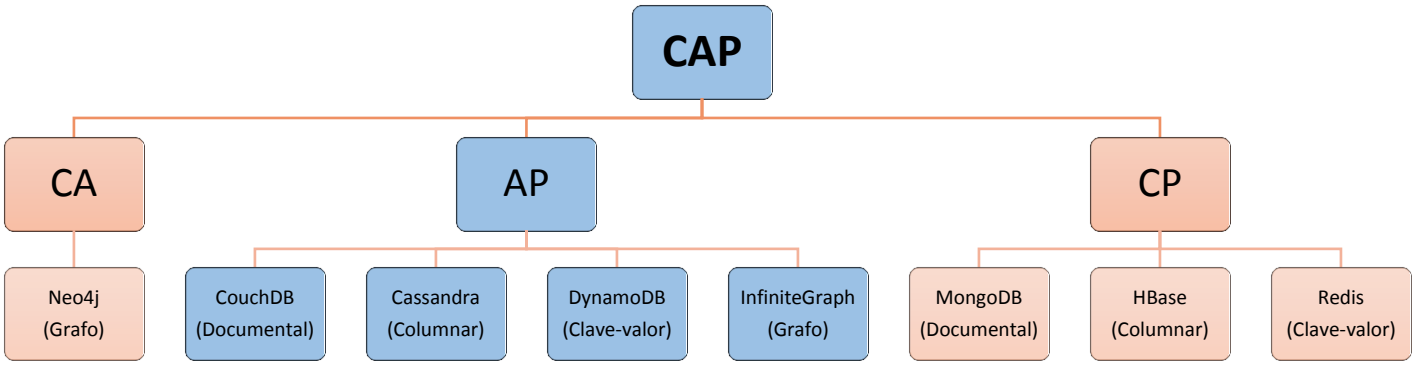

Ilustración 8: Árbol CAP de decisión de la solución

Si usamos el segundo árbol, vamos a decantarnos por la base de datos tipo Columnar. Esto es así porque los datos que tenemos tienen una estructura similar a la usada por este tipo de sistemas. Cada usuario tiene muchos atributos, pero algunos no los tienen todos, por lo que la estructura de los datos sería algo similar a lo que aparece en la *ILUSTRACIÓN 1: CÓMO ALMACENAN LA INFORMACIÓN LAS BBDD NOSQL*.

En la siguiente imagen se muestra el árbol de decisión tipo BBDD, con el camino elegido marcado en azul. De este camino salen 2 opciones: Cassandra y HBase.

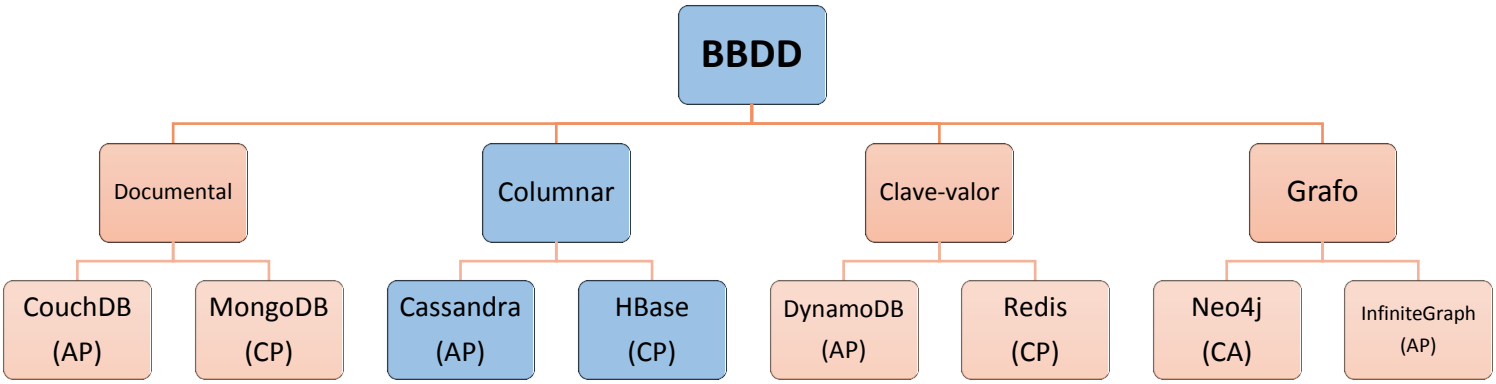

Ilustración 9: Árbol BBDD de decisión de la solución

Para decantarnos por uno de ellos, cruzamos los resultados salidos del primer árbol de decisión con los resultados salidos del segundo, y vemos que la única opción que aparece en ambos es **Cassandra**. Sin embargo, debemos asegurarnos de que este sistema cumple con los requerimientos establecidos en *3.3 – CARACTERÍSTICAS DE LA SOLUCIÓN DESEADA*.

CARACTERÍSTICAS	
El sistema debe permitir insertar y actualizar usuarios, tanto de uno en uno a través de la interfaz como cargando un archivo a la base de datos.	✓
El sistema debe permitir borrar un usuario o todos los que haya en la base de datos.	✓
El sistema debe permitir la búsqueda de usuarios por cualquiera de sus atributos.	✓
La interfaz gráfica del sistema debe mostrar el resultado de las consultas realizadas por el usuario.	✓
El sistema debe permitir la exportación de la información desde la base de datos a un archivo.	✓
El sistema debe restringir con unas credenciales las peticiones de inserción, actualización y borrado.	✓
El sistema debe permitir dar de alta, de baja o modificar los permisos de acceso de un usuario.	✓
El sistema debe permitir las conexiones remotas.	✓
El sistema debe poder ser implantado en los equipos actuales del Laboratorio del Departamento de Informática.	✓
El sistema debe poder escalarse horizontalmente.	✓
El sistema debe ser de código abierto y licencia gratuita.	✓
El mantenimiento del sistema debe ser poco costoso en horas por persona.	✓

Tabla 2: Características cumplidas por Cassandra

La justificación de que Cassandra cumple con las 8 primeras características viene explicada en *6.3 – MODO DE USO*.

Cassandra es fácilmente instalable en cualquier equipo, incluido en los del Laboratorio, puesto que su único requisito es tener instalado Java en el equipo, algo que la mayoría de ellos ya poseen; y para los que no, viene incluido en los pasos de instalación de Cassandra. Además, funciona tanto en sistemas operativos Windows como en Linux, esto nos permite en un trabajo futuro poder usarlo también en Windows.

La escalabilidad horizontal es una de las características básicas de los sistemas NoSQL, como ya se explicó en *2.1 – INTRODUCCIÓN*.

Aunque inicialmente Cassandra fue desarrollada por Facebook, posteriormente se convirtió en un proyecto de código abierto de Apache, por lo que ahora puede descargarse gratuitamente de su página web [28].

El mantenimiento de esta base de datos es muy bajo, ya que una vez que los datos estén almacenados y las políticas de acceso establecidas, no habrá que revisar el sistema a menos de que surja algún error o se quiera modificar algún dato.

3.5 – Requisitos

Estos requisitos se han obtenido de una entrevista con Roberto Fuentes Astorga, uno de los administradores del actual sistema de almacenamiento LDAP. Tras la entrevista, se definieron los requisitos y se volvió a concertar una reunión con el cliente, con el fin de revisar que los requisitos recogieran todas las funcionalidades que se pedían para el sistema y corregir aquellas que pudieran causar ambigüedad o duda.

3.5.1 – Requisitos de usuario

Los requisitos indican la funcionalidad que va a ofrecer el sistema y las restricciones que se le impondrán. Cada requisito de usuario indica una funcionalidad o restricción del sistema, y puede englobar varias funcionalidades en el mismo requisito. Los requisitos de usuario se dividen en dos grupos:

- **Requisitos de capacidad**. Indican las funciones y operaciones que ofrece el sistema.
- **Requisitos de restricción**. Indican las restricciones impuestas sobre el sistema.

Cada requisito de usuario viene definido en una tabla como la siguiente:

Identificador			
Título			
Fuente	☐ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☐ Deseable	☐ Opcional
Prioridad	☐ Alta	☐ Media	☐ Baja
Verificabilidad	☐ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☐ Inestable	
Descripción			

Tabla 3: Plantilla para requisitos de usuario

Cada requisito de usuario contendrá los campos a continuación descritos:

- **Identificador**. Cada requisito tendrá un identificador propio del tipo RUX-YY para diferenciarlo de los demás, donde:
  - **RU**. Indica que el requisito es de usuario.
  - **X**. Indica el tipo de requisito de usuario, que puede ser:
    - **C**. Requisito de capacidad.
    - **R**. Requisito de restricción.
  - **YY**. Indica el número de requisito, que va de 01 a 99.
- **Título**. Breve descripción única del requisito.
- **Fuente**. El origen del requisito, pudiendo ser el cliente o el desarrollador.
- **Necesidad**. Necesidad del requisito por parte del cliente. Puede tomar tres valores:
  - Esencial. El requisito es necesario para el funcionamiento del sistema.
  - Deseable. El requisito se encuentra dentro de los objetivos del desarrollador, pero puede ser aplazado a versiones posteriores del sistema.
  - Opcional. El requisito no es necesario para el funcionamiento del sistema y puede omitirse en versiones posteriores.
- **Prioridad**. Importancia que tiene la funcionalidad especificada por el requisito a la hora de implementarlo. Puede tomar tres valores: Alta, Media o Baja.
- **Verificabilidad**. Posibilidad de comprobar si el software emplea el requisito y cumple el objetivo para el que está designado. Puede tomar tres valores: Alta, Media o Baja.
- **Estabilidad**. Posibilidad de que el requisito tenga que ser cambiado en versiones posteriores. Puede tomar dos valores:
  - Estable. No es probable que se modifique el requisito.
  - Inestable. Es probable que se modifique el requisito.
- **Descripción**. Explicación detallada del requisito.

3.5.1.1 – Requisitos de capacidad

En este apartado se enumeran los requisitos de capacidad, que indican las funciones del sistema.

RUC-01			
Título	El sistema permitirá almacenar usuarios, modificar sus atributos y eliminar a los mismos, tanto de forma individual como múltiple.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que se inserten, actualicen o borren usuarios tanto de uno en uno como todos a la vez. Al insertar y actualizar, se podrá hacer introduciendo los datos de un usuario, o importando un archivo con múltiples usuarios. Al borrar, se podrá borrar un solo usuario, o la base de datos entera.		

Tabla 4: Requisito de usuario RUC-01

RUC-02			
Título	El sistema permitirá buscar a los usuarios por cualquier atributo.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que se realicen consultas sobre cualquier atributo que exista en el modelo.		

Tabla 5: Requisito de usuario RUC-02

RUC-03			
Título	El sistema permitirá guardar los datos en un archivo.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá exportar la información de la base de datos a un archivo externo que servirá como copia de seguridad.		

Tabla 6: Requisito de usuario RUC-03

RUC-04			
Título	El sistema mostrará la ayuda para la elección de una base de datos NoSQL.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema debe contar con un sistema de ayuda para que el usuario pueda elegir entre otros tipos de bases de datos NoSQL distintas a la propuesta, por si encontrara una que se ajuste más a sus necesidades.		

Tabla 7: Requisito de usuario RUC-04

RUC-05			
Título	El sistema permitirá crear, modificar y borrar un rol.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador cree, modifique y borre un rol.		

Tabla 8: Requisito de usuario RUC-05

RUC-06			
Título	El sistema permitirá asignar o eliminar los permisos de un rol.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador asigne o elimine los permisos de un rol.		

Tabla 9: Requisito de usuario RUC-06

RUC-07			
Título	El sistema permitirá listar los roles y sus permisos asociados.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador liste tanto los roles como los permisos asociados a ellos.		

Tabla 10: Requisito de usuario RUC-07

RUC-08			
Título	La inicialización del sistema no contará con más de 3 comandos.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	No se deberán de realizar más de 3 comandos para inicializar el sistema y que esté disponible para consultar la información.		

Tabla 11: Requisito de usuario RUC-08

3.5.1.2 – Requisitos de restricción

En este apartado se enumeran los requisitos de restricción, que indican las restricciones del sistema.

RUR-01			
Título	El sistema permitirá realizar búsquedas a cualquier usuario, pero sólo el administrador podrá almacenar, modificar y borrar datos.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que cualquier usuario pueda realizar una consulta al sistema. A la hora de realizar inserciones, actualizaciones y borrados, el sistema sólo permitirá su realización si se tienen los permisos para ello.		

Tabla 12: Requisito de usuario RUR-01

RUR-02			
Título	El sistema tardará menos de 5 minutos en recuperarse de un fallo.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☐ Media	☒ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Ante un fallo en el software del sistema, se tardará menos de 5 minutos en restaurar los datos importándolos desde un archivo de recuperación.		

Tabla 13: Requisito de usuario RUR-02

RUR-03			
Título	El tiempo de respuesta del sistema no será mayor que 5 segundos.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El tiempo de respuesta por una petición no será en ningún caso superior a 5 segundos.		

Tabla 14: Requisito de usuario RUR-03

RUR-04			
Título	Se podrá acceder al sistema de forma remota.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá realizar operaciones de forma remota desde cualquier dirección IP. Antes de su ejecución, al igual que en local, se pedirán credenciales para realizar dichas operaciones.		

Tabla 15: Requisito de usuario RUR-04

RUR-05			
Título	La ayuda para la elección de una base de datos debe incluir una documentación.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	La ayuda para la elección de una base de datos debe incluir una documentación sobre las características de cada tipo específico y de cada opción elegible, para que los usuarios inexpertos tengan toda la información necesaria para realizar su elección.		

Tabla 16: Requisito de usuario RUR-05

RUR-06			
Título	El sistema contará con un solo superusuario.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema contará con un solo rol que podrá ejercer de superusuario, pudiendo realizar todas las acciones posibles sobre el sistema sin ningún tipo de restricción.		

Tabla 17: Requisito de usuario RUR-06

RUR-07			
Título	Para usar el sistema será obligatorio iniciar sesión.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	Para usar el sistema será necesario iniciar sesión con un usuario y una contraseña previamente definidas, no pudiendo acceder a él sin posesión de estas credenciales.		

Tabla 18: Requisito de usuario RUR-07

RUR-08			
Título	Sólo el superusuario podrá gestionar los roles.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá crear, modificar y borrar un rol.		

Tabla 19: Requisito de usuario RUR-08

RUR-09			
Título	Sólo el superusuario podrá gestionar los permisos de un rol.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá asignar o eliminar permisos a un rol.		

Tabla 20: Requisito de usuario RUR-09

RUR-10			
Título	Sólo el superusuario podrá ver los roles y permisos.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá listar los roles y los permisos asociados a estos.		

Tabla 21: Requisito de usuario RUR-10

3.5.2 – Requisitos de software

Los requisitos de software describen de forma detallada cada una de las funcionalidades englobadas en un requisito de usuario. A continuación, voy a describir los requisitos de software, los cuales son derivados de los requisitos de usuario anteriormente definidos. Estos requisitos se dividen en dos grupos:

- **Requisitos funcionales**. Indican el funcionamiento esencial del sistema.
- **Requisitos no funcionales**. Indican funcionalidad añadidas del sistema que no son necesarias para su esencial funcionamiento.

Cada requisito de software vendrá definido en una tabla como la siguiente:

Identificador			
Título			
Fuente	☐ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☐ Deseable	☐ Opcional
Prioridad	☐ Alta	☐ Media	☐ Baja
Verificabilidad	☐ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☐ Inestable	
Descripción			
Procedencia			

Tabla 22: Plantilla para requisitos de software

Cada requisito de software contendrá los campos a continuación descritos:

- **Identificador**. Cada requisito tendrá un identificador propio del tipo RSX-YY para diferenciarlo de los demás, donde:
  - **RS**. Indica que el requisito es de software.
  - **X**. Indica el tipo de requisito de software, que puede ser:
    - **F**. Requisito funcional.
    - **N**. Requisito no funcional.
  - **YY**. Indica el número de requisito, que va de 01 a 99.
- **Título**. Breve descripción única del requisito.
- **Fuente**. El origen del requisito, pudiendo ser el cliente o el desarrollador.
- **Necesidad**. Necesidad del requisito por parte del cliente. Puede tomar tres valores:
  - Esencial. El requisito es necesario para el funcionamiento del sistema.
  - Deseable. El requisito se encuentra dentro de los objetivos del desarrollador, pero puede ser aplazado a versiones posteriores del sistema.
  - Opcional. El requisito no es necesario para el funcionamiento del sistema y puede omitirse en versiones posteriores.
- **Prioridad**. Importancia que tiene la funcionalidad especificada por el requisito a la hora de implementarlo. Puede tomar tres valores: Alta, Media o Baja.
- **Verificabilidad**. Posibilidad de comprobar si el software emplea el requisito y cumple el objetivo para el que está designado. Puede tomar tres valores: Alta, Media o Baja.
- **Estabilidad**. Posibilidad de que el requisito tenga que ser cambiado en versiones posteriores. Puede tomar dos valores:
  - Estable. No es probable que se modifique el requisito.
  - Inestable. Es probable que se modifique el requisito.
- **Descripción**. Explicación detallada del requisito.
- **Procedencia**. Requisito de usuario en el que se basa el requisito de software.

3.5.2.1 – Requisitos funcionales

En este apartado se enumeran los requisitos funcionales, que indican el funcionamiento del sistema.

RSF-01			
Título	El sistema permitirá realizar la inserción de un usuario.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá realizar la inserción de los datos de un usuario.		
Procedencia	RUC-01		

Tabla 23: Requisito de software RSF-01

RSF-02			
Título	El sistema permitirá realizar la inserción de múltiples usuarios.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá realizar la inserción de múltiples usuarios importando los datos desde un fichero externo.		
Procedencia	RUC-01		

Tabla 24: Requisito de software RSF-02

RSF-03			
Título	El sistema permitirá la actualización de un usuario.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá realizar la actualización de los datos de un usuario.		
Procedencia	RUC-01		

Tabla 25: Requisito de software RSF-03

RSF-04			
Título	El sistema permitirá la actualización de múltiples usuarios.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá realizar la actualización de múltiples usuarios importando los datos desde un fichero externo.		
Procedencia	RUC-01		

Tabla 26: Requisito de software RSF-04

RSF-05			
Título	El sistema permitirá el borrado de un usuario.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá realizar el borrado de un usuario indicando la información del usuario a eliminar.		
Procedencia	RUC-01		

Tabla 27: Requisito de software RSF-05

RSF-06			
Título	El sistema permitirá el borrado de múltiples usuarios.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá realizar el borrado de toda la información contenida en la base de datos.		
Procedencia	RUC-01		

Tabla 28: Requisito de software RSF-06

RSF-07			
Título	El sistema permitirá realizar consultas por cualquier atributo.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que se realicen consultas sobre cualquier atributo que exista en el modelo.		
Procedencia	RUC-02		

Tabla 29: Requisito de software RSF-07

RSF-08			
Título	El sistema permitirá exportar los datos a un archivo.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá exportar la información de la base de datos a un archivo externo que servirá como copia de seguridad.		
Procedencia	RUC-03		

Tabla 30: Requisito de software RSF-08

RSF-09			
Título	El sistema mostrará la ayuda para la elección de una base de datos NoSQL.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema mostrará la ayuda para la elección de una base de datos NoSQL cuando el usuario la solicite.		
Procedencia	RUC-04		

Tabla 31: Requisito de software RSF-09

RSF-10			
Título	El sistema permitirá crear un rol.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador cree un rol.		
Procedencia	RUC-05		

Tabla 32: Requisito de software RSF-10

RSF-11			
Título	El sistema permitirá modificar un rol.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador modifique un rol.		
Procedencia	RUC-05		

Tabla 33: Requisito de software RSF-11

RSF-12			
Título	El sistema permitirá borrar un rol.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador borre un rol.		
Procedencia	RUC-05		

Tabla 34: Requisito de software RSF-12

RSF-13			
Título	El sistema permitirá asignar permisos a un rol.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador asigne diferentes permisos a un rol, como el de superusuario, inicio de sesión, lectura o escritura sobre la base de datos.		
Procedencia	RUC-06		

Tabla 35: Requisito de software RSF-13

RSF-14			
Título	El sistema permitirá eliminar los permisos de un rol.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador elimine los permisos de un rol.		
Procedencia	RUC-06		

Tabla 36: Requisito de software RSF-14

RSF-15			
Título	El sistema permitirá listar los roles.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador liste los roles almacenados.		
Procedencia	RUC-07		

Tabla 37: Requisito de software RSF-15

RSF-16			
Título	El sistema permitirá listar los permisos de un rol.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá que el administrador liste los permisos asociados a un rol.		
Procedencia	RUC-07		

Tabla 38: Requisito de software RSF-16

RSF-17			
Título	La inicialización del sistema no contará con más de 3 comandos.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	Para inicializar el sistema, sólo será necesario crear un *keyspace*, crear una tabla e importar los datos a la tabla creada. De este modo el sistema ya está listo para ser usado.		
Procedencia	RUC-08		

Tabla 39: Requisito de software RSF-17

3.5.2.2 – Requisitos no funcionales

En este apartado se enumeran los requisitos no funcionales, que indican funcionalidades añadidas del sistema.

RSN-01			
Título	La exportación de los datos se realizará una vez al mes.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☐ Media	☒ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Una vez al mes se exportará la información de la base de datos a un archivo externo para actualizar la copia de seguridad, por si en ese mes hubiera habido alguna actualización de los datos.		
Procedencia	RUC-03		

Tabla 40: Requisito de software RSN-01

RSN-02			
Título	El sistema de ayuda se basará en un árbol de decisión.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema de ayuda será un árbol de decisión. El usuario irá recorriendo sus distintas ramas según lo que vaya eligiendo de entre las opciones que se le presentan.		
Procedencia	RUC-04		

Tabla 41: Requisito de software RSN-02

RSN-03			
Título	El sistema de ayuda tendrá dos métodos de elección.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema de ayuda tendrá dos métodos de elección: uno basado en el Teorema CAP, y otro basado en el tipo de base de datos NoSQL.		
Procedencia	RUC-04		

Tabla 42: Requisito de software RSN-03

RSN-04			
Título	El árbol de decisión partirá de un tipo general y derivará en varios tipos específicos.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El árbol de decisión tendrá como origen un tipo general (Teorema CAP o tipo de base de datos NoSQL), del cual se irá dividiendo en varios tipos específicos.		
Procedencia	RUC-04		

Tabla 43: Requisito de software RSN-04

RSN-05			
Título	El árbol de decisión proporcionará varias opciones por cada tipo específico.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El árbol de decisión proporcionará varias opciones por cada tipo específico para que el usuario pueda elegir la que más le guste o se adapte a la funcionalidad que necesite.		
Procedencia	RUC-04		

Tabla 44: Requisito de software RSN-05

RSN-06			
Título	El sistema permitirá realizar consultas a cualquier usuario.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema no restringirá las consultas, pudiendo ser realizadas por cualquier usuario.		
Procedencia	RUR-01		

Tabla 45: Requisito de software RSN-06

RSN-07			
Título	El sistema permitirá realizar inserciones sólo al administrador.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema restringirá la inserción de datos pidiendo unas credenciales, de modo que sólo el administrador pueda realizar esta operación.		
Procedencia	RUR-01		

Tabla 46: Requisito de software RSN-07

RSN-08			
Título	El sistema permitirá realizar actualizaciones sólo al administrador.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema restringirá la actualización de datos pidiendo unas credenciales, de modo que sólo el administrador pueda realizar esta operación.		
Procedencia	RUR-01		

Tabla 47: Requisito de software RSN-08

RSN-09			
Título	El sistema permitirá realizar borrados sólo al administrador.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema restringirá el borrado de datos pidiendo unas credenciales, de modo que sólo el administrador pueda realizar esta operación.		
Procedencia	RUR-01		

Tabla 48: Requisito de software RSN-09

RSN-10			
Título	El sistema tardará menos de 5 minutos en recuperarse de un fallo de software.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☐ Media	☒ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Ante un fallo en el software del sistema, se tardará menos de 5 minutos en solucionarlo e importar los datos desde el archivo de copia de seguridad realizado cada mes.		
Procedencia	RUR-02		

Tabla 49: Requisito de software RSN-10

RSN-11			
Título	El tiempo de respuesta del sistema no será mayor que 5 segundos.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El tiempo de respuesta por una petición no será en ningún caso superior a 5 segundos.		
Procedencia	RUR-03		

Tabla 50: Requisito de software RSN-11

RSN-12			
Título	El sistema permitirá la conexión desde cualquier dirección IP.		
Fuente	☒ Cliente	☐ Desarrollador	
Necesidad	☒ Esencial	☐ Deseable	☐ Opcional
Prioridad	☒ Alta	☐ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema permitirá realizar operaciones de forma remota desde cualquier dirección IP. Antes de su ejecución, al igual que pasa en local, se pedirán credenciales para realizar dichas operaciones.		
Procedencia	RUR-04		

Tabla 51: Requisito de software RSN-12

RSN-13			
Título	Los dos métodos de elección vendrán explicados junto al árbol de decisión.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	La ayuda de elección de base de datos deberá explicar la diferencia entre los distintos tipos específicos, así como qué características garantiza y no garantiza cada tipo.		
Procedencia	RUR-05		

Tabla 52: Requisito de software RSN-13

RSN-14			
Título	La ayuda de elección de base de datos explicará las características de cada opción proporcionada por el árbol de decisión.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☐ Alta	☒ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	La ayuda de elección de base de datos explicará las características de cada opción proporcionada por el árbol de decisión junto al mismo, para que el usuario conozca bien la base de datos NoSQL que va a elegir.		
Procedencia	RUR-05		

Tabla 53: Requisito de software RSN-14

RSN-15			
Título	El sistema contará con un solo superusuario.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	El sistema contará con un solo rol que podrá ejercer de superusuario, pudiendo realizar todas las acciones posibles sobre el sistema sin ningún tipo de restricción.		
Procedencia	RUR-06		

Tabla 54: Requisito de software RSN-15

RSN-16			
Título	Para usar el sistema será necesario iniciar sesión.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☒ Estable	☐ Inestable	
Descripción	Para usar el sistema será necesario iniciar sesión con un usuario y una contraseña, que el sistema contrastará con los almacenados en los ficheros de configuración para permitir o no el acceso.		
Procedencia	RUR-07		

Tabla 55: Requisito de software RSN-16

RSN-17			
Título	Sólo el superusuario podrá crear un rol.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá crear un rol.		
Procedencia	RUR-08		

Tabla 56: Requisito de software RSN-17

RSN-18			
Título	Sólo el superusuario podrá modificar un rol.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá modificar un rol.		
Procedencia	RUR-08		

Tabla 57: Requisito de software RSN-18

RSN-19			
Título	Sólo el superusuario podrá borrar un rol.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá borrar un rol.		
Procedencia	RUR-08		

Tabla 58: Requisito de software RSN-19

RSN-20			
Título	Sólo el superusuario podrá asignar permisos a un rol.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá asignar permisos a un rol.		
Procedencia	RUR-09		

Tabla 59: Requisito de software RSN-20

RSN-21			
Título	Sólo el superusuario podrá eliminar los permisos de un rol.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá eliminar los permisos de un rol.		
Procedencia	RUR-09		

Tabla 60: Requisito de software RSN-21

RSN-22			
Título	Sólo el superusuario podrá ver los roles.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá listar los roles existentes en el sistema.		
Procedencia	RUR-10		

Tabla 61: Requisito de software RSN-22

RSN-23			
Título	Sólo el superusuario podrá ver los permisos de un rol.		
Fuente	☐ Cliente	☒ Desarrollador	
Necesidad	☐ Esencial	☒ Deseable	☐ Opcional
Prioridad	☐ Alta	☒ Media	☐ Baja
Verificabilidad	☒ Alta	☐ Media	☐ Baja
Estabilidad	☐ Estable	☒ Inestable	
Descripción	Sólo el rol con permisos de superusuario podrá listar los permisos asociados a los distintos roles existentes en el sistema.		
Procedencia	RUR-10		

Tabla 62: Requisito de software RSN-23

3.6 – Matrices de trazabilidad

Una matriz de trazabilidad sirve para verificar requisitos, ya sea respecto a su procedencia, respecto a la funcionalidad del sistema que garantizan, o respecto a las pruebas realizadas.

3.6.1 – Matriz de trazabilidad entre requisitos de usuario y requisitos de software

A continuación se encuentra la matriz de trazabilidad, que verifica que todos los requisitos de software parten de un requisito de usuario.

		REQUISITOS DE USUARIO																
		RUC-01	RUC-02	RUC-03	RUC-04	RUC-05	RUC-06	RUC-07	RUC-08	RUR-01	RUR-02	RUR-03	RUR-04	RUR-05	RUR-06	RUR-07	RUR-08	RUR-09	RUR-10
REQUISITOS DE SOFTWARE	RSF-01	✓																	
	RSF-02	✓																	
	RSF-03	✓																	
	RSF-04	✓																	
	RSF-05	✓																	
	RSF-06	✓																	
	RSF-07		✓																
	RSF-08			✓															
	RSF-09				✓														
	RSF-10					✓													
	RSF-11					✓													
	RSF-12					✓													
	RSF-13						✓												
	RSF-14						✓												
	RSF-15							✓											
	RSF-16							✓											
	RSF-17								✓										
	RSN-01			✓															
	RSN-02				✓														
	RSN-03				✓														
	RSN-04				✓														
	RSN-05				✓														
	RSN-06									✓									
	RSN-07									✓									
	RSN-08									✓									
	RSN-09									✓									
	RSN-10										✓								
	RSN-11											✓							
	RSN-12												✓						
	RSN-13													✓					
	RSN-14													✓					
	RSN-15														✓				
	RSN-16															✓			
	RSN-17																✓		
	RSN-18																✓		
	RSN-19																✓		
	RSN-20																	✓	
	RSN-21																	✓	
	RSN-22																		✓
	RSN-23																		✓

Tabla 63: Matriz de trazabilidad entre requisitos de usuario y requisitos de software

3.7 – Entorno operacional

Como se ha comentado anteriormente, el sistema debe poder ser implantado en los equipos del Laboratorio del Departamento de Informática. Las especificaciones técnicas de estos equipos son las siguientes:

ESPECIFICACIONES TÉCNICAS	
Sistema Operativo	Windows 7 Professional 64bits
Procesador	Intel Core i5-4460 3.2GHz
Memoria RAM	16 GB
Disco duro	1 TB

Tabla 64: Especificaciones técnicas de los equipos del Laboratorio

Para este proyecto, el entorno operacional es una máquina virtual a la que se le han asignado las siguientes especificaciones técnicas:

ESPECIFICACIONES TÉCNICAS	
Sistema Operativo	Ubuntu Desktop x64 (versión 16.04.1)
Memoria RAM	5 GB
Disco duro	50 GB

Tabla 65: Especificaciones técnicas de la máquina virtual

Las especificaciones técnicas del equipo en el que se ha montado la máquina virtual son las siguientes:

ESPECIFICACIONES TÉCNICAS	
Sistema Operativo	Windows 10 Pro 64bits (versión 1607)
Procesador	Intel Core i7-5500U 2.4GHz
Memoria RAM	8 GB
Disco duro	1 TB

Tabla 66: Especificaciones técnicas de mi equipo

En el caso de que hubiera un acceso remoto al equipo en el que se encuentra la base de datos, la comunicación entre ambos equipos, local y remoto, quedaría descrita por la siguiente imagen, tanto si es un acceso desde la red de la Universidad como si es un acceso desde fuera.

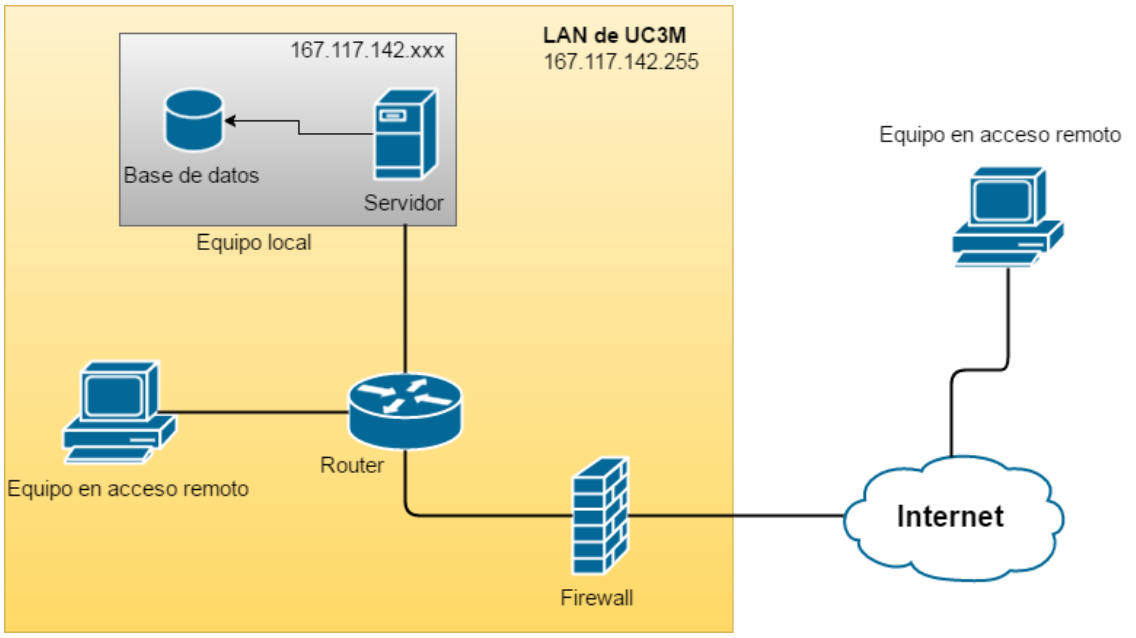

Ilustración 10: Diagrama de red

Como podemos observar, la red de la Universidad cuenta con una dirección IP pública, y por tanto accesible desde Internet. Sin embargo, cuenta con un Firewall en la entrada para defender la red frente a posibles ataques como DDoS. Adicionalmente, también puede darse el caso de un acceso remoto desde la misma red de la Universidad. Este acceso sería más rápido y fluido puesto que el equipo local y el remoto están en la misma red.

4 – Diseño

Este apartado se va a dividir en 3 partes. En la primera se explica cuál es la arquitectura que se ha diseñado para el sistema y qué diseño tiene la base de datos; en la segunda se muestra, con diagramas de casos de uso, de clase y de secuencia, el flujo de información y las posibilidades que nos ofrece el sistema; y en la tercera y última se hace una revisión de la interfaz del sistema.

Aunque anteriormente ya se explicó por qué se ha escogido una base de datos NoSQL como solución final, antes de ponerme a explicar el diseño de la solución escogida, voy a hacer un pequeño resumen de lo que ya expliqué con detalle en el apartado *2.2 – ESTUDIO DE POSIBLES SOLUCIONES*.

La principal razón por la que esta solución se ha impuesto sobre una base de datos relacional, que hay que admitir era la opción más viable de entre las tres posibles soluciones estudiadas, es que implementa las mismas funcionalidades que una BDR, pero cuenta con cosas que ésta no posee, como la posibilidad de cambiar fácilmente el modelo de datos o la escalabilidad horizontal sobre la vertical. Además, el hecho de que el NoSQL sea un tema relativamente novedoso en estos tiempos me hizo decantarme por estudiar este tema, ya que es algo que no he visto en ningún momento del Grado. Por último, uno de los objetivos del Trabajo Fin de Grado es ser capaz de adquirir conocimientos sobre un área de manera autónoma, mediante una incisiva búsqueda de información, para luego ser capaz de transmitir esos conocimientos a otras personas, demostrando así el nivel que se posee del tema en cuestión. Por lo tanto, creo que realizar el Trabajo Fin de Grado sobre un tema que desconocía totalmente cumple bastante bien con ese objetivo.

4.1 – Definición de la arquitectura del sistema

Dado que el sistema es una base de datos, su arquitectura es del tipo ANSI-SPARC. Esta arquitectura se divide en 3 niveles [29]:

- **Nivel Externo** (para usuarios). Visión de la base de datos según cada usuario. Está representado por un Esquema Externo, que permite que los usuarios vean sólo la parte de la base de datos que les interesa.
- **Nivel Medio** (para diseñadores). Visión global de la estructura de los datos. Está representado por un Esquema Conceptual, que sirve de punto de control para futuros desarrollos de la base de datos y aísla la representación de la información de las necesidades de la máquina y las exigencias de los usuarios.
- **Nivel Interno** (para SSOO). Registros almacenados. Está representado por un Esquema Interno, que especifica qué son y cómo se almacenan los datos.

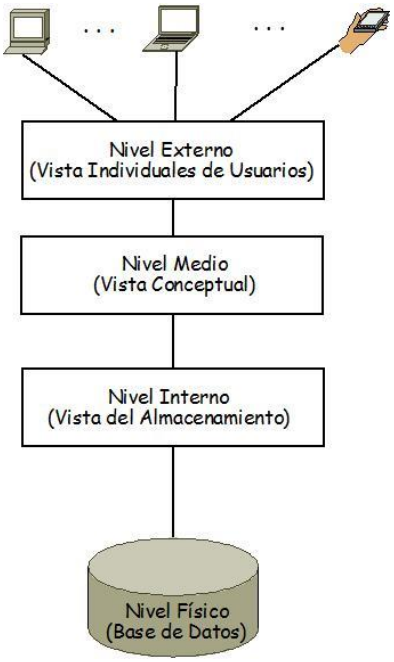

Ilustración 11: Arquitectura ANSI-SPARC

Además de la arquitectura ANSI-SPARC, Cassandra posee las siguientes características propias [30]:

- **Es distribuida**. Cassandra almacena la información de manera replicada en varios nodos. De este modo, puede garantizar que los datos estén siempre disponibles.
- **Escala lineal y horizontalmente**. Cassandra cuenta con escalabilidad lineal, que permite aumentar la capacidad de cómputo del sistema exponencialmente si añadimos más nodos, y escalabilidad horizontal, que permite aumentar el rendimiento del sistema si añadimos más hardware a los nodos.
- **Implementa una arquitectura P2P**. Además de ANSI-SPARC, Cassandra implementa una arquitectura Peer-to-peer (ver *APÉNDICE I – ACRÓNIMOS Y DEFINICIONES*), por lo que no sigue los patrones maestro-esclavo de otros sistemas de almacenamiento, y evita el fallo en cascada porque un nodo maestro pueda fallar.

4.2 – Diseño de la base de datos

En este apartado se muestra cuál es el diseño de la base de datos y cómo se va a almacenar la información de los miembros del Personal del Departamento de Informática.

Después de inspeccionar la página web del Personal del Dpto. de Informática, e ir viendo qué información se almacenaba de cada miembro, he podido comprobar que se guarda la información relativa a los siguientes campos: nombre, apellidos, email, teléfono, campus, edificio, despacho, categoría, área de conocimiento. Sin embargo, existen ciertas personas que no poseen todos estos atributos. Por tanto, al ser Cassandra una base de datos de tipo columnar, es perfecta para almacenar los datos del Personal del Dpto. de Informática, ya que los campos rellenos no van a ser iguales para todos los usuarios.

Cassandra es una base de datos no relacional, por lo que no sigue el Modelo Entidad-Relación ni el Modelo Relacional. De este modo, su diseño no es igual que el de los sistemas relacionales, que se basan en un esquema relacional. Cassandra sigue un modelo de datos basado en columnas, que cuenta con los siguientes elementos: Columna, Supercolumna, Fila o Registro, Familia de columnas, Espacio Clave y Clúster [31] [32]. En el siguiente diagrama se ve gráficamente este modelo aplicado a los datos del personal de la universidad:

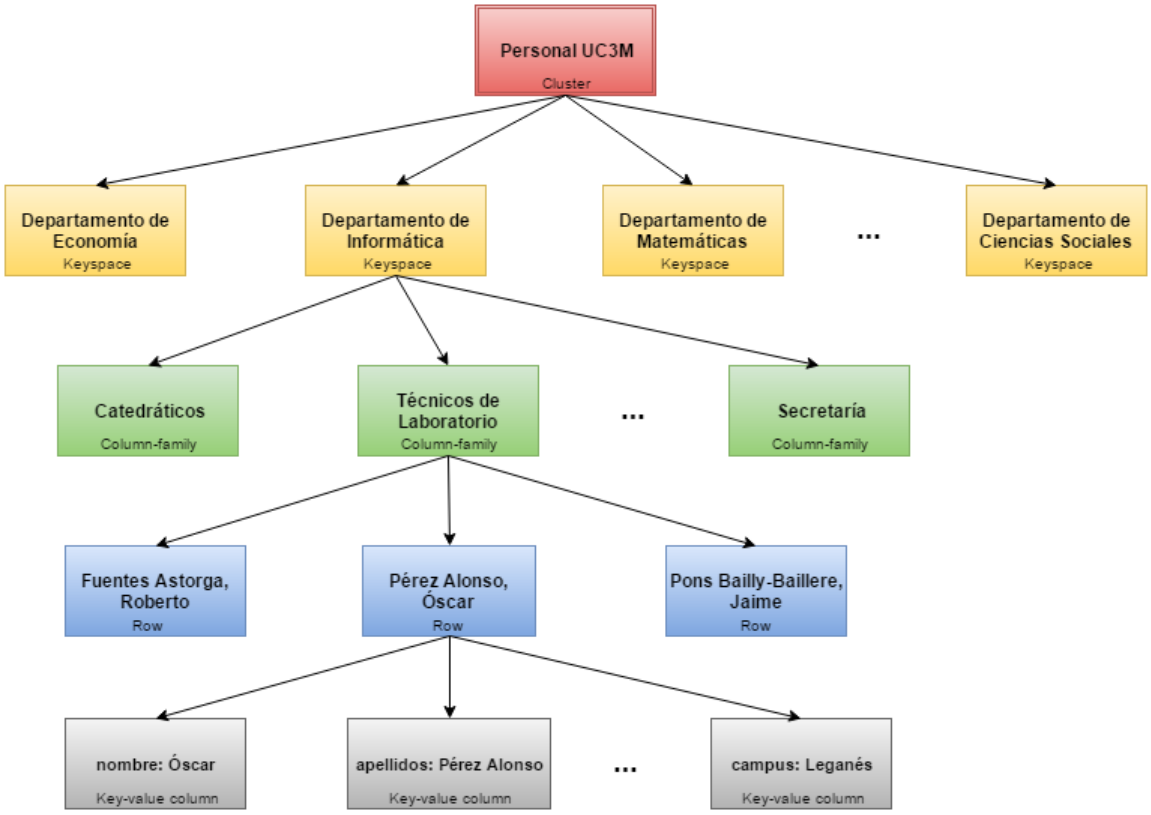

Ilustración 12: Diagrama de la base de datos según el modelo de datos de Cassandra

Como podemos ver, el sistema estará compuesto por un *Cluster*, donde se almacena la información. En él tenemos varios *Keyspaces*, cada uno para un departamento de la universidad (en este caso sólo hemos implementado el del Dpto. de Informática, el resto lo dejamos como trabajo futuro). Cada *Keyspace* contendrá a su vez varias *Column-families*, cada una asociada a un grupo del departamento, y cada *Column-familiy* almacenará varias *Rows*, cada una con un miembro de ese grupo. Por último, cada *Row* contendrá varias *Key-value column*, cada una con un dato de ese miembro, como nombre o email.

4.3 – Diseño de casos de uso

En este apartado se especifican los distintos casos en los que un usuario va a interaccionar con el sistema, los cuales surgen de la definición de los *3.5.2 – REQUISITOS DE SOFTWARE* mencionada anteriormente. Estos casos se definen como Casos de Uso, según lo establecido en Métrica v3.

Cada caso de uso vendrá definido por:

1. Un esquema a modo de explicación gráfica llamado diagrama de caso de uso, basado en UML (ver *APÉNDICE I – ACRÓNIMOS Y DEFINICIONES*), y desarrollado por medio de la herramienta Draw.io [26]. El siguiente diagrama representa todos los casos de uso existentes.

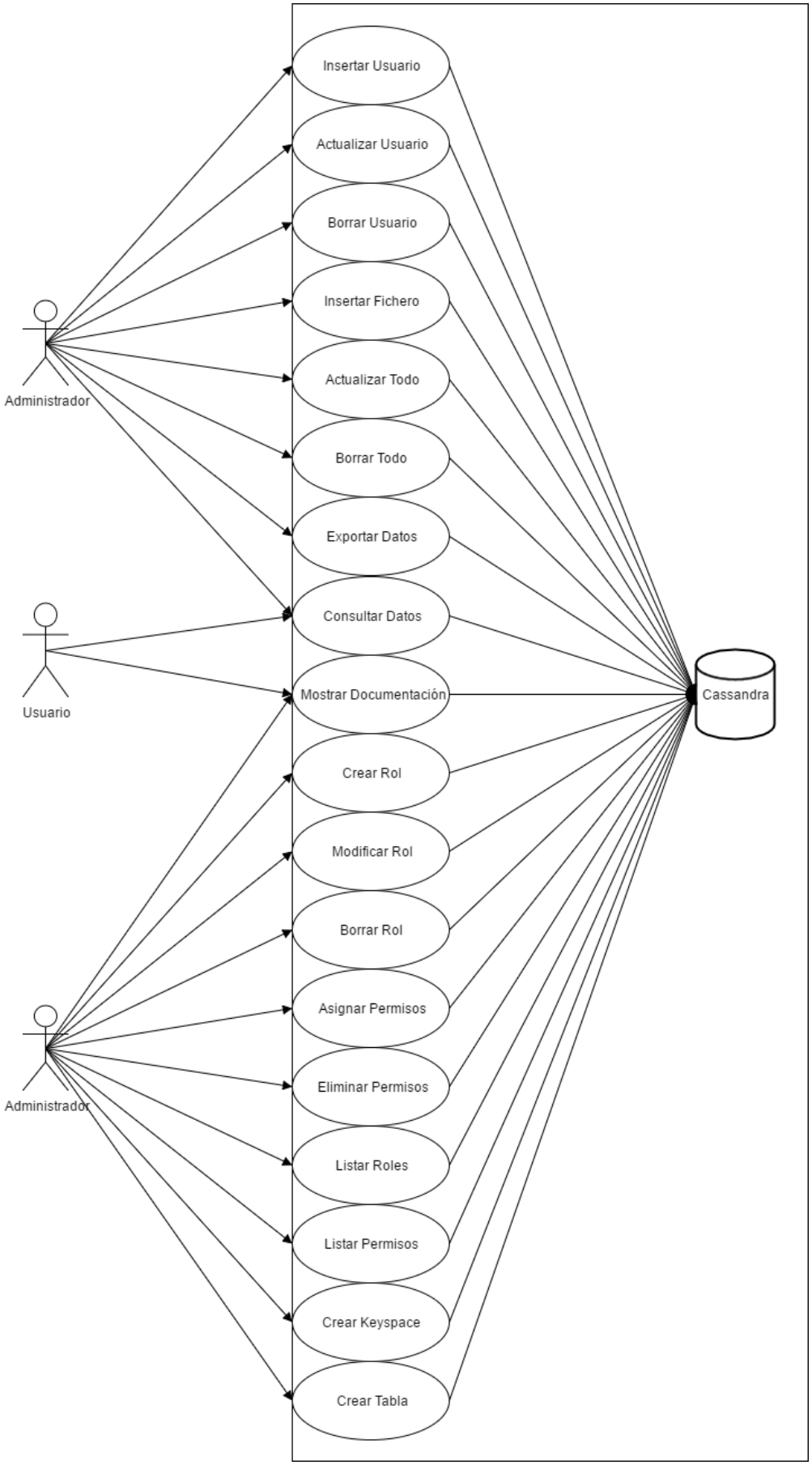

Ilustración 13: Casos de Uso

2. Una tabla como la siguiente:

Identificador	
Título	
Actor principal	
Objetivo	
Escenario principal	
Precondiciones	
Postcondiciones	
Frecuencia	

Tabla 67: Plantilla para casos de uso

Cada caso de uso contendrá los campos a continuación descritos:

- **Identificador**. Cada caso de uso tendrá un identificador propio del tipo CU-XX para diferenciarlo de los demás, donde:
  - **CU**. Indica que es un caso de uso.
  - **XX**. Indica el número de caso de uso, que va de 01 a 99.
- **Título**. Nombre único del caso de uso.
- **Actor principal**. El rol que interactúa con el sistema. Es externo al sistema desarrollado. Puede tomar dos valores:
  - **Usuario**. El actor es un usuario cualquiera, sin privilegios de ningún tipo.
  - **Administrador**. El actor es un usuario responsable de la administración del sistema, como puede ser un técnico de laboratorio, que tiene acceso privilegiado al mismo.
- **Objetivo**. El propósito del caso de uso.
- **Escenario principal**. Los pasos que debe realizar el actor principal para completar el objetivo.
- **Precondiciones**. Los requisitos que deben cumplirse para llevar a cabo el escenario.
- **Postcondiciones**. El estado final del sistema tras llevar a cabo el escenario.
- **Frecuencia**. Frecuencia con la que se realiza el caso de uso.

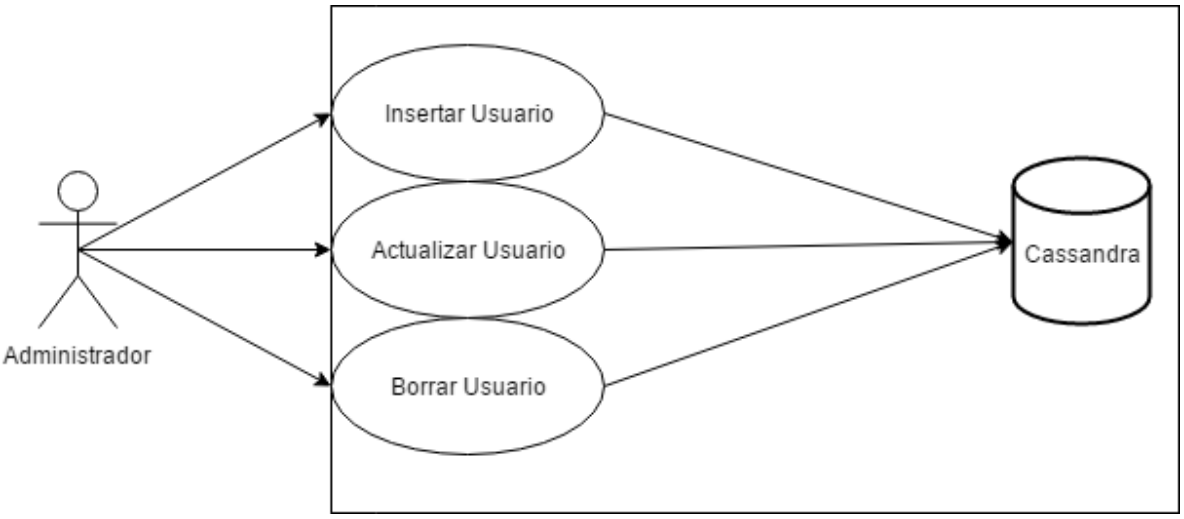

Ilustración 14: Diagrama de caso de uso CU-01

CU-01	
Título	Inserción/Actualización/Borrado de un usuario.
Actor principal	Administrador.
Objetivo	Insertar/Actualizar/Borrar un usuario en la base de datos.
Escenario principal	El administrador ingresa a Cassandra con su usuario y contraseña, se coloca en el *keyspace* adecuado y ejecuta el comando INSERT/UPDATE/DELETE.
Precondiciones	Haber iniciado sesión con un usuario y una contraseña válidos, haberse colocado en el *keyspace* correcto, que la tabla esté creada y que el usuario exista (actualizar/borrar).
Postcondiciones	Se insertará/actualizará/borrará un usuario nuevo en la base de datos.
Frecuencia	Cuando se añada un miembro al Personal del Departamento de Informática/Cuando cambie su información/Cuando se elimine un miembro del Personal.

Tabla 68: Caso de uso CU-01

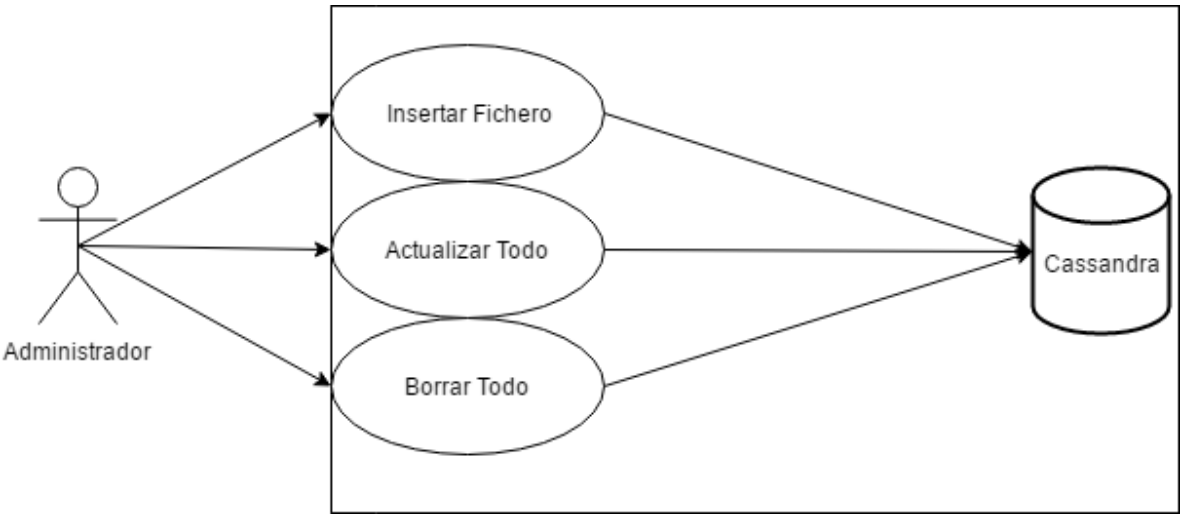

Ilustración 15: Diagrama de caso de uso CU-02

CU-02	
Título	Inserción/Actualización/Borrado de muchos usuarios.
Actor principal	Administrador.
Objetivo	Insertar/Actualizar a la vez muchos usuarios o borrar todos los usuarios en la base de datos.
Escenario principal	El administrador ingresa a Cassandra con su usuario y contraseña, se coloca en el *keyspace* adecuado y ejecuta el comando COPY FROM/COPY FROM/TRUNCATE.
Precondiciones	Haber iniciado sesión con un usuario y una contraseña válidos, haberse colocado en el *keyspace* correcto, que la tabla esté creada y que el archivo exista (insertar/actualizar).
Postcondiciones	Se insertarán/actualizarán los datos del archivo o se eliminará toda la información de la base de datos.
Frecuencia	Cuando se inicie la base de datos por primera vez y cuando se restablezca tras un fallo/Cuando cambie la información de muchos miembros del Personal del Departamento de Informática en muy poco tiempo/Cuando se corrompa la base de datos y haya que reestablecer la información.

Tabla 69: Caso de uso CU-02

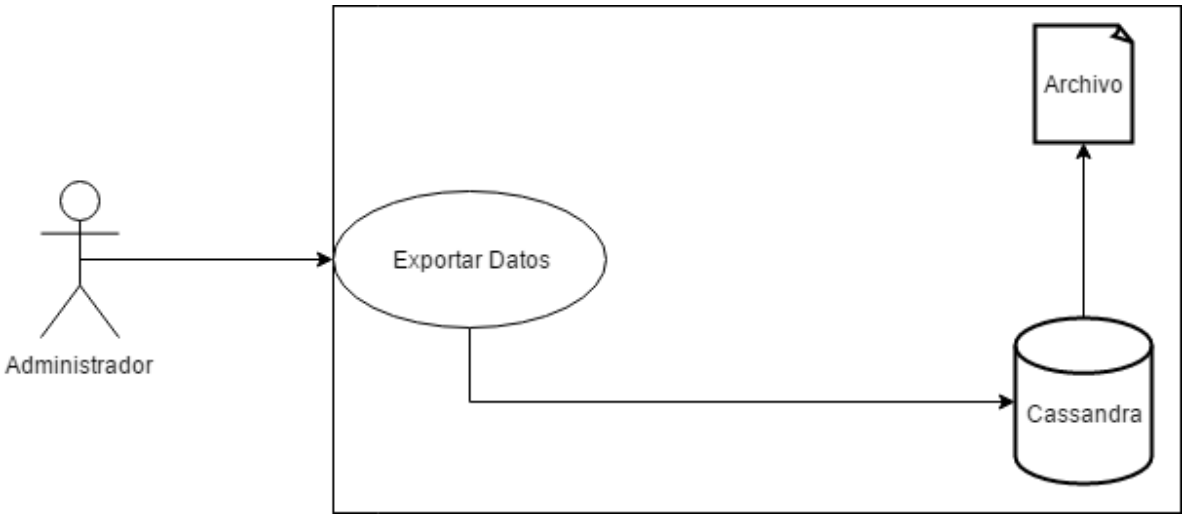

Ilustración 16: Diagrama de caso de uso CU-03

CU-03	
Título	Exportación de los usuarios a un archivo.
Actor principal	Administrador.
Objetivo	Exportar la información de la base de datos a un archivo.
Escenario principal	El administrador ingresa a Cassandra con su usuario y contraseña, se coloca en el *keyspace* adecuado y ejecuta el comando COPY TO.
Precondiciones	Haber iniciado sesión con un usuario y una contraseña válidos, haberse colocado en el *keyspace* correcto, que la tabla esté creada y que el archivo exista.
Postcondiciones	Se guardará la información de la base de datos en un archivo.
Frecuencia	Una vez al mes, como copia de seguridad.

Tabla 70: Caso de uso CU-03

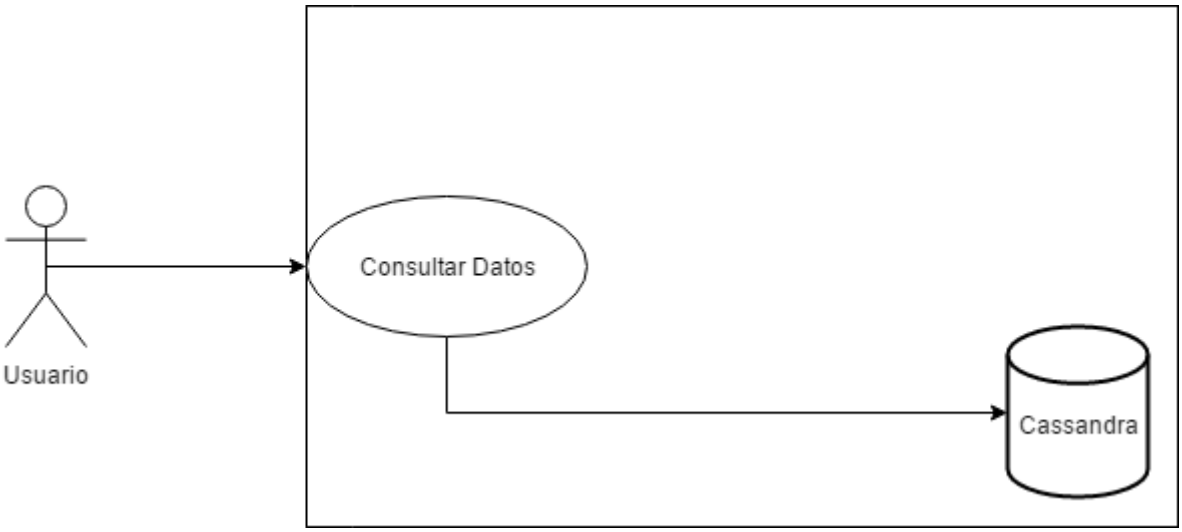

Ilustración 17: Diagrama de caso de uso CU-04

CU-04	
Título	Consulta de uno o varios usuarios.
Actor principal	Usuario.
Objetivo	Consultar la información de uno o varios usuarios de la base de datos.
Escenario principal	El usuario ingresa a Cassandra con su usuario y contraseña, se coloca en el *keyspace* adecuado y ejecuta el comando SELECT.
Precondiciones	Haber iniciado sesión con un usuario y una contraseña válidos, haberse colocado en el *keyspace* correcto y que la tabla esté creada.
Postcondiciones	Se recuperará la información de uno o varios usuarios.
Frecuencia	Cuando un usuario quiera acceder a la información de algún miembro del Personal del Departamento de Informática.

Tabla 71: Caso de uso CU-04

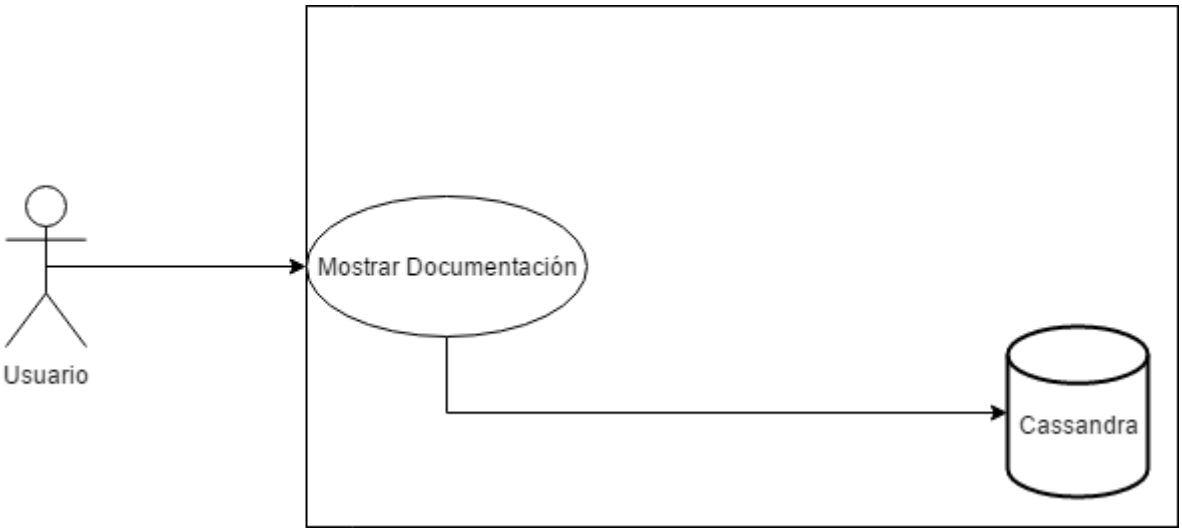

Ilustración 18: Diagrama de caso de uso CU-05

CU-05	
Título	Mostrar la ayuda de elección de base de datos.
Actor principal	Usuario.
Objetivo	Hacer uso de la ayuda de elección de base de datos.
Escenario principal	El usuario ejecuta el método SHOW_DOC para acceder a la ayuda de elección de base de datos (dicho método no pertenece a Cassandra, por lo que debe ser creado; su creación se deja como trabajo futuro).
Precondiciones	Que el archivo HTML esté creado.
Postcondiciones	Se mostrará la ayuda de elección de base de datos.
Frecuencia	Cuando un usuario quiera acceder a la ayuda de elección de base de datos.

Tabla 72: Caso de uso CU-05

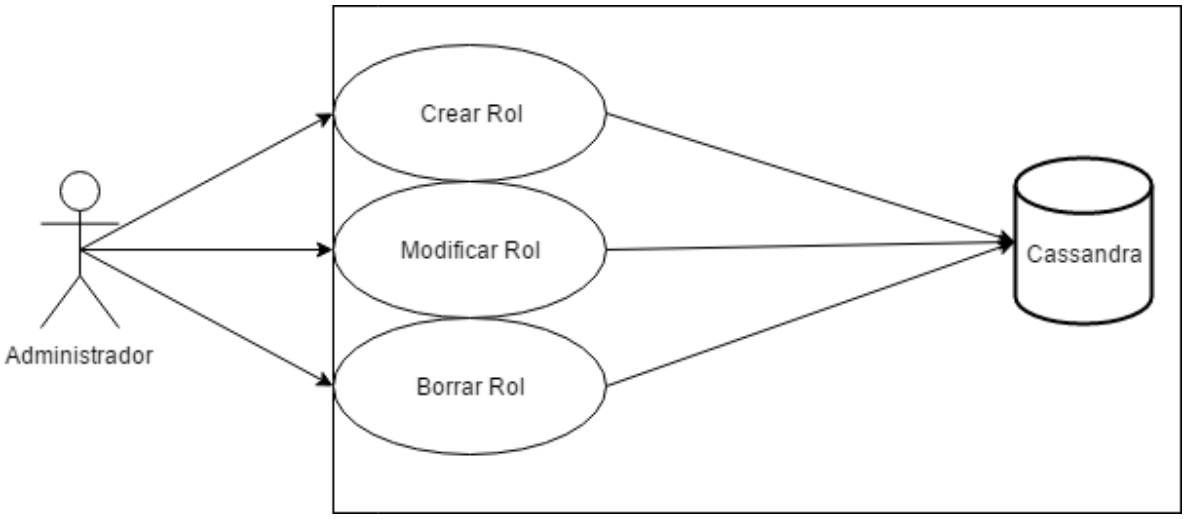

Ilustración 19: Diagrama de caso de uso CU-06

CU-06	
Título	Creación/Modificación/Borrado de un rol.
Actor principal	Administrador.
Objetivo	Crear/Modificar/Borrar un rol en la base de datos.
Escenario principal	El administrador ingresa a Cassandra con su usuario y contraseña y ejecuta el comando CREATE/ALTER/DROP ROLE.
Precondiciones	Haber iniciado sesión con un usuario y una contraseña válidos, haber modificado previamente el archivo de configuración para habilitar la autenticación y autorización y que el rol exista (actualizar/borrar).
Postcondiciones	Se añadirá/modificará/eliminará un rol de la base de datos.
Frecuencia	Cuando se creen los roles Administrador y Usuario, y cuando se quiera añadir otro rol con privilegios para ayudar en el mantenimiento de la base de datos/Cada vez que se quieran modificar las propiedades de un rol para darle más o menos privilegios/Cuando se quiera dar de baja un rol.

Tabla 73: Caso de uso CU-06

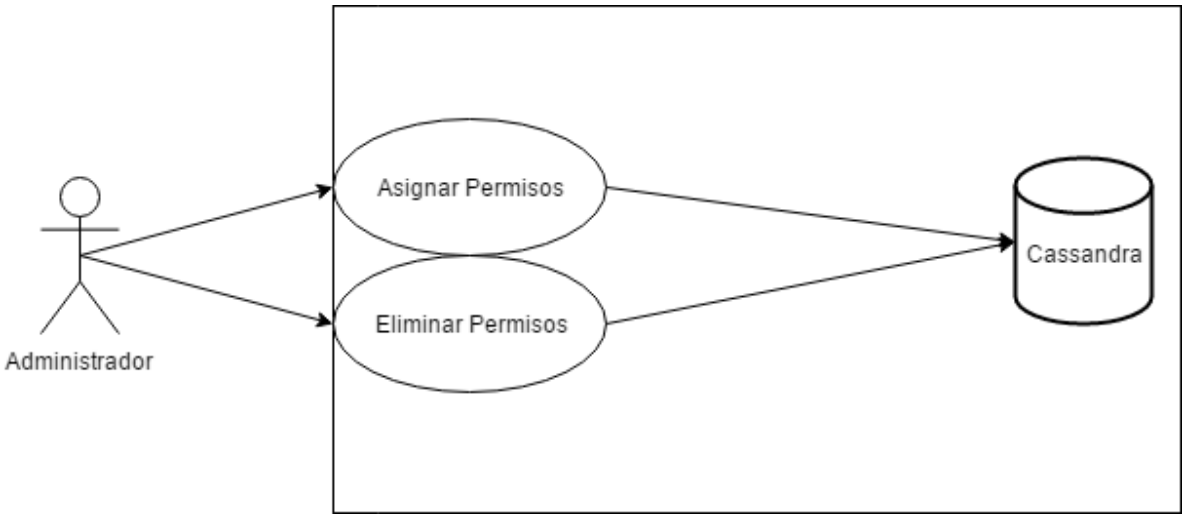

Ilustración 20: Diagrama de caso de uso CU-07

CU-07	
Título	Asignación/Eliminación de permisos a un rol.
Actor principal	Administrador.
Objetivo	Asignar/Eliminar permisos a un rol.
Escenario principal	El administrador ingresa a Cassandra con su usuario y contraseña y ejecuta el comando GRANT/REVOKE.
Precondiciones	Haber iniciado sesión con un usuario y una contraseña válidos, haber modificado previamente el archivo de configuración para habilitar la autenticación y autorización y que el rol exista.
Postcondiciones	Se asignarán/eliminarán permisos a un rol.
Frecuencia	Cada vez que se le quiera dar más/menos privilegio a un rol.

Tabla 74: Caso de uso CU-07

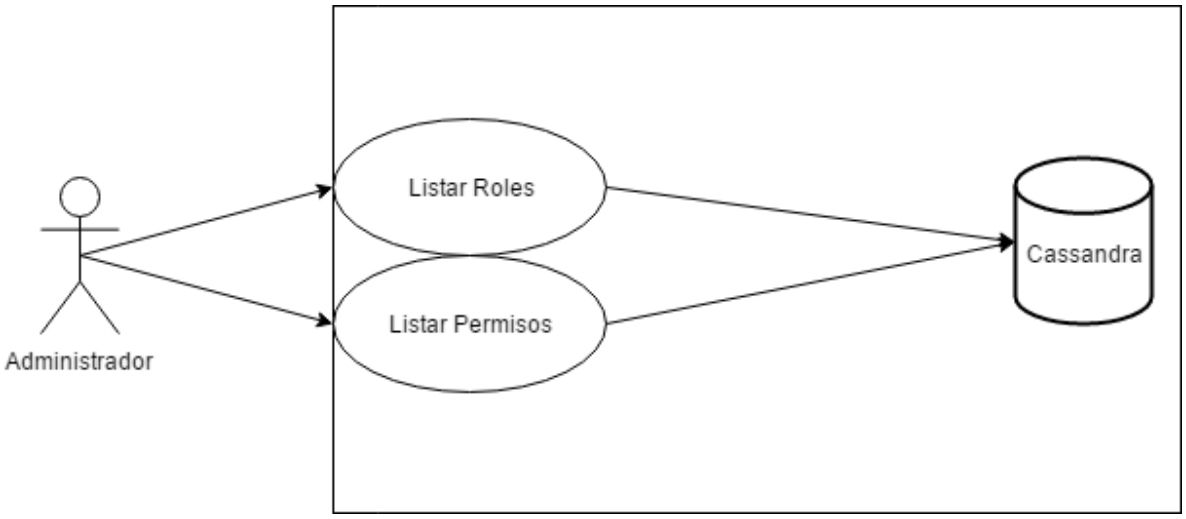

Ilustración 21: Diagrama de caso de uso CU-08

CU-08	
Título	Listado de los roles/permisos de un rol.
Actor principal	Administrador.
Objetivo	Listar los roles/permisos de un rol existentes en la base de datos.
Escenario principal	El administrador ingresa a Cassandra con su usuario y contraseña y ejecuta el comando LIST ROLES/LIST ALL PERMISSIONS OF.
Precondiciones	Haber iniciado sesión con un usuario y una contraseña válidos, haber modificado previamente el archivo de configuración para habilitar la autenticación y autorización. y que el rol exista (listar permisos).
Postcondiciones	Se mostrarán todos los roles de la base de datos/permisos de un rol.
Frecuencia	Cada vez que se quieran ver los roles de la base de datos/permisos de un rol.

Tabla 75: Caso de uso CU-08

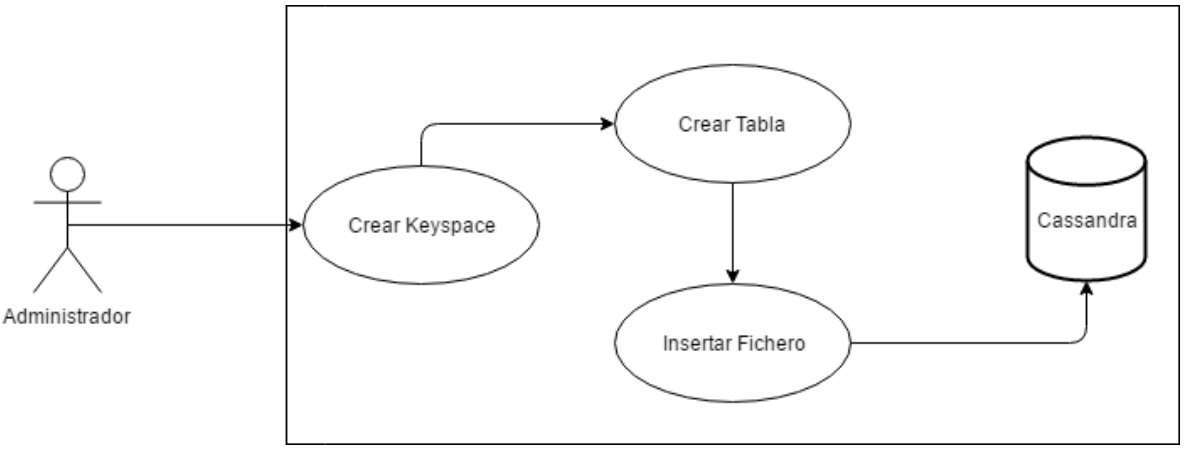

Ilustración 22: Diagrama de caso de uso CU-09

CU-09	
Título	Inicializar sistema
Actor principal	Administrador.
Objetivo	Inicializar el sistema para que esté disponible para su uso.
Escenario principal	El administrador ingresa a Cassandra y ejecuta los comandos CREATE KEYSPACE, CREATE TABLE y COPY FROM.
Precondiciones	Que el archivo con los datos exista y tenga formato JSON o CSV.
Postcondiciones	Se tendrá un nuevo *keyspace* y una nueva tabla rellena de datos.
Frecuencia	Después de instalar el sistema.

Tabla 76: Caso de uso CU-09

4.4 – Diseño de clases

Una vez realizado el diseño de los casos de uso, toca hacer lo propio con las clases que derivan de dichos casos. El siguiente esquema, basado en UML y realizado por medio de la herramienta Draw.io [26], representa las clases existentes en el sistema:

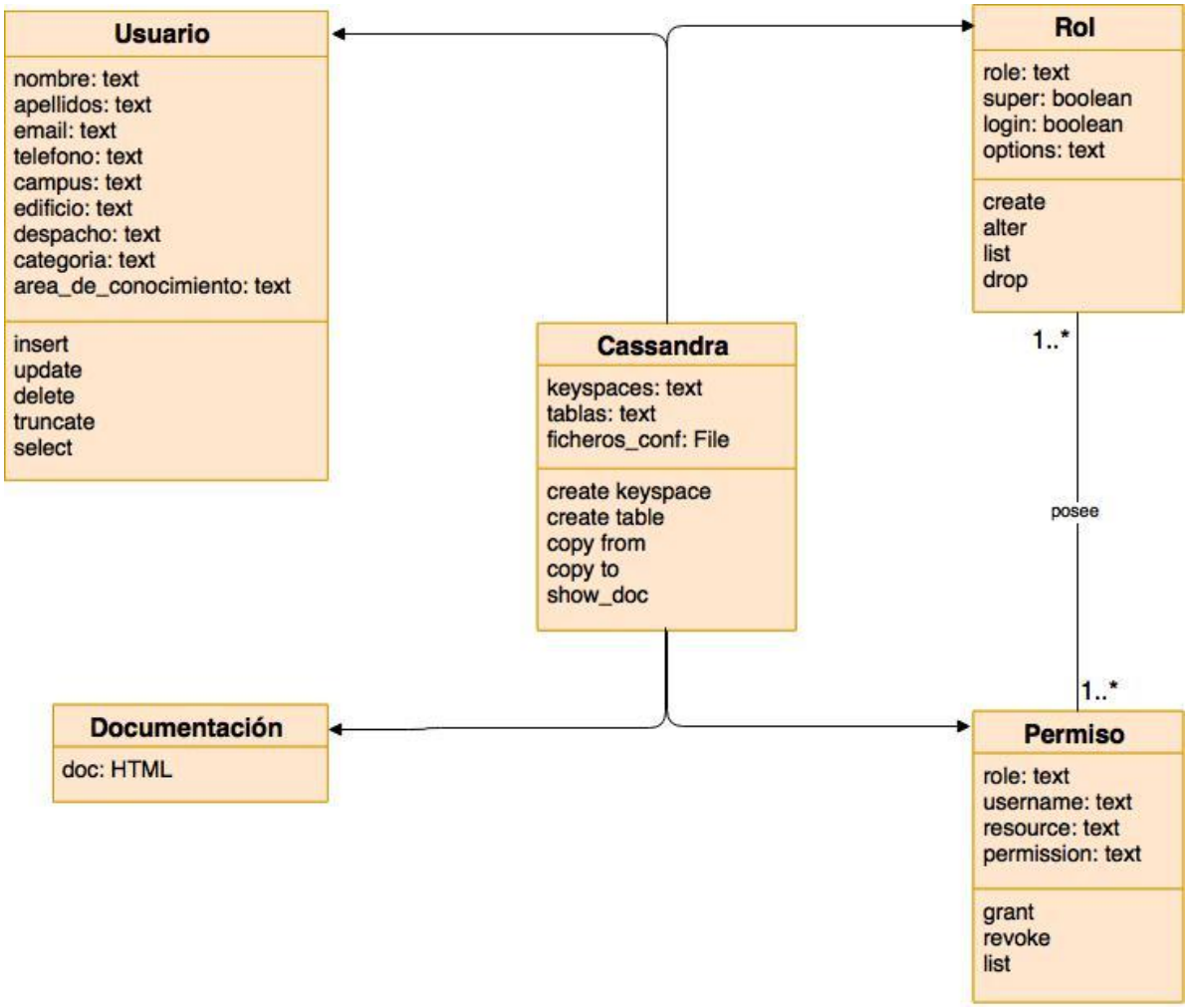

Ilustración 23: Diagrama de clases

A continuación se va a explicar detalladamente cada una de estas clases. La primera de ellas es la clase Cassandra, que puede considerarse la principal del sistema por ser la que gestiona los datos, los permisos y los roles. Esta clase contiene información sobre los *keyspaces* y tablas creadas, así como de los ficheros de configuración del sistema. También contiene los métodos asociados al sistema, como crear *keyspace* y tabla, importar y exportar datos, o mostrar la documentación de ayuda de elección de una base de datos. Esta clase se relaciona con la clase Documentación mediante un método que accede y muestra el HTML que contiene la ayuda de elección de la base de datos, con la clase Usuarios ejecutando los métodos de dicha clase para realizar operaciones sobre la base de datos, con la clase Rol ejecutando los métodos de esa clase para gestionar los roles del sistema, y con la clase Permiso ejecutando sus métodos para gestionar los permisos de un rol.

Clase Cassandra	
create keyspace	
Objetivo	Crear un *keyspace* para almacenar tablas.
Argumentos	Nombre del *keyspace* y parámetros de replicación.
Retorno	-
create table	
Objetivo	Crear una tabla para almacenar datos.
Argumentos	Nombre del *keyspace* donde se va a crear la tabla, nombre de la tabla y nombre y tipo de dato de cada columna de la tabla, indicando qué columna ejerce de clave primaria.
Retorno	-
copy from	
Objetivo	Importar la información de un archivo a la base de datos.
Argumentos	Tabla donde meter los datos, atributos de la tabla y ruta donde se encuentra el fichero.
Retorno	Número de filas importadas, número de archivos de origen y tiempo de ejecución de la petición.
copy to	
Objetivo	Exportar la información de la base de datos a un archivo.
Argumentos	Tabla donde están los datos, atributos de la tabla y ruta donde se encuentra el fichero.
Retorno	Número de filas exportadas, número de archivos de destino y tiempo de ejecución de la petición.
show_doc	
Objetivo	Mostrar la ayuda de elección de base de datos.
Argumentos	-
Retorno	HTML donde se encuentra la documentación de ayuda.

Tabla 77: Clase Cassandra

La clase Documentación únicamente contiene el archivo HTML con la ayuda de elección de base de datos.

La clase Usuario representa los datos almacenados en el sistema. Esta clase contiene todos los posibles atributos que pueden tener los usuarios, así como los métodos relacionados con la gestión de los datos del sistema, ya sea almacenar, modificar, borrar o buscar un usuario.

Clase Usuario	
insert	
Objetivo	Insertar un nuevo usuario en el sistema.
Argumentos	Tabla en la que se quiere introducir el usuario, atributos que se quieren introducir y el valor de esos atributos.
Retorno	-
update	
Objetivo	Actualizar la información de un usuario del sistema.
Argumentos	Tabla donde se encuentra el usuario, atributos que se quieren actualizar, nuevo valor para esos atributos e información del usuario para buscarlo.
Retorno	-
delete	
Objetivo	Borrar un usuario del sistema.
Argumentos	Tabla donde se encuentra el usuario e información del usuario para buscarlo.
Retorno	-
truncate	
Objetivo	Borrar todos los usuarios del sistema.
Argumentos	Tabla que se quiere borrar.
Retorno	-
select	
Objetivo	Consultar la información de un usuario del sistema.
Argumentos	Atributos que se quieren recuperar del usuario, tabla donde se encuentra el usuario e información del usuario para buscarlo.
Retorno	Atributos elegidos de los usuarios que contengan la información pasada en el argumento.

Tabla 78: Clase Usuario

La clase Rol existe para gestionar el acceso al sistema. Esta clase contiene la información asociada al rol, así como los métodos relacionados con la gestión de roles, ya sea crear, modificar, listar o eliminar. Esta clase se relaciona con la clase Permiso de tal modo que un rol puede tener uno o muchos permisos asociados.

Clase Rol	
create	
Objetivo	Crear un nuevo rol en el sistema.
Argumentos	Nombre del rol, contraseña del rol y, opcionalmente, valores de superuser y login.
Retorno	-
alter	
Objetivo	Modificar un rol del sistema.
Argumentos	Nombre del rol, contraseña del rol y valor de superuser y/o login.
Retorno	-
list	
Objetivo	Ver los roles del sistema.
Argumentos	-
Retorno	Roles del sistema, con su nombre y propiedades.
drop	
Objetivo	Eliminar un rol del sistema.
Argumentos	Nombre del rol.
Retorno	-

Tabla 79: Clase Rol

La clase Permiso existe para gestionar el acceso a los datos del sistema. Esta clase contiene la información asociada a los permisos que un rol tiene sobre el acceso a la información almacenada en el sistema, así como los métodos relacionados para gestionar esos permisos, ya sea agregar o eliminar. Esta clase se relaciona con la clase Rol de tal modo que un permiso sobre un dato puede estar asociado a uno o muchos roles.

Clase Permiso	
grant	
Objetivo	Añadirle un permiso a un rol.
Argumentos	Tipo de permiso, recurso donde ejecutar el permiso y nombre del rol.
Retorno	-
revoke	
Objetivo	Eliminar un permiso de un rol.
Argumentos	Tipo de permiso, recurso donde se tenía el permiso y nombre del rol.
Retorno	-
list	
Objetivo	Ver los permisos de un rol.
Argumentos	Nombre del rol.
Retorno	Rol elegido, usuario del rol, recursos y permiso sobre cada recurso.

Tabla 80: Clase Permiso

4.5 – Diseño de secuencia

Por último, tras realizar el diseño de casos de uso y el diseño de clases, sólo queda representar el diagrama de secuencia. En este diagrama se puede ver la interacción entre un usuario y el sistema, con el fin de realizar los casos de usos especificados, ejecutando los métodos definidos en cada una de las clases indicadas en dicho diagrama.

Para no recargar el documento, se ha decido separar el diagrama de secuencia en varios trozos, para ir explicándolos uno a uno.

El primero de ellos es el diagrama de secuencia del inicio de sesión. Aunque no se ha definido como un caso de uso, pues su ejecución es algo obligatorio para poder realizar los casos de uso definidos, el inicio de sesión es una forma de interacción con el sistema, por tanto, he visto necesario añadirlo al diagrama de secuencia. El resultado de esta interacción puede ser doble: por un lado, si las credenciales que introducimos existen y cuentan con el permiso de hacer login, Cassandra no nos devolverá ningún valor, pero nos dejará acceder al sistema; sin embargo, si dichas credenciales no existen o no cuentan con el permiso de hacer login, no nos dejará acceder al sistema y nos devolverá un mensaje indicándonos que ese usuario y contraseña no tienen permiso para acceder. Para comprobar las credenciales, Cassandra contrasta los datos introducidos por el usuario con los datos almacenados en los ficheros de configuración.

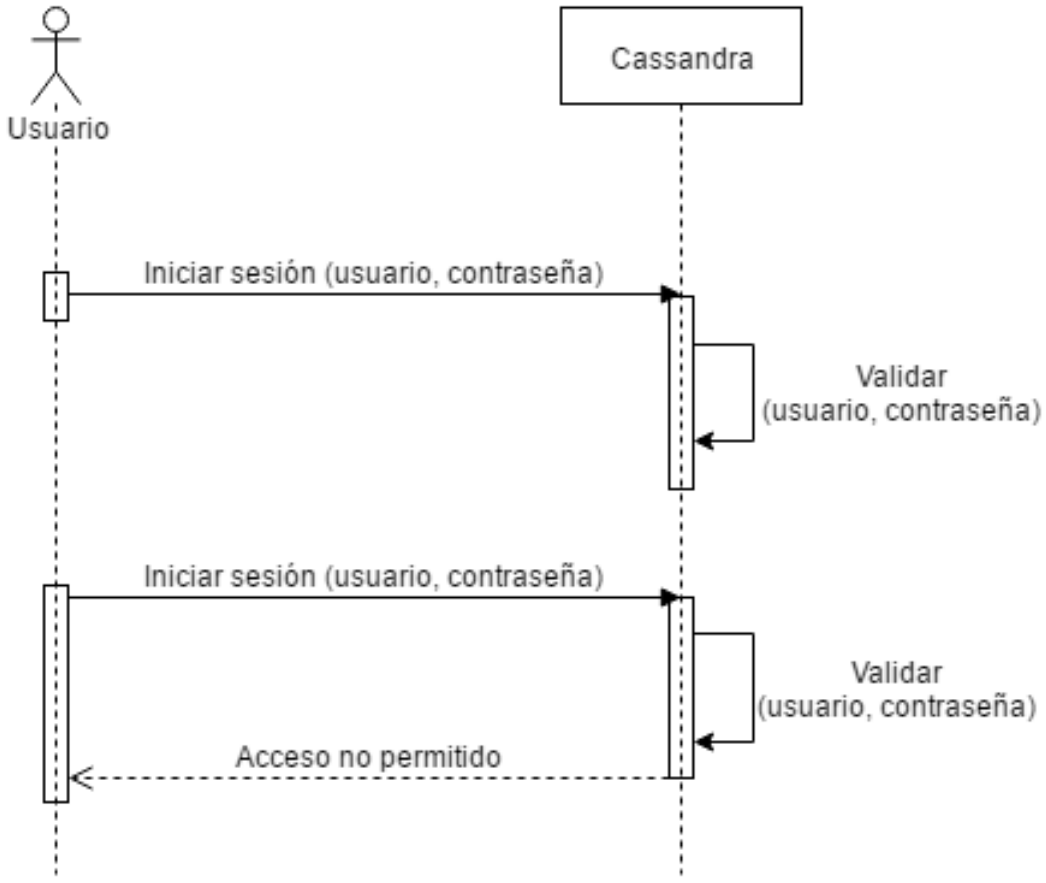

Ilustración 24: Diagrama de secuencia de inicio de sesión

En los siguientes diagramas se da por entendido que el usuario tiene permiso para hacer login, por lo que no se especificará en el diagrama el resultado de que no tuviera dicho permiso, puesto que ya se ha especificado en el diagrama de inicio de sesión.

El siguiente representa los métodos contenidos en la clase Cassandra. Los métodos *Create Keyspace* y *Create Table* no reciben respuesta, y si el usuario tiene permiso para ejecutar esos comandos, Cassandra creará un nuevo *keyspace* y una nueva tabla con el nombre y propiedades que reciba del usuario. Los métodos *Copy From* y *Copy To* reciben el número de filas que se han importado a la base de datos desde el archivo o exportado al archivo desde la base de datos, el archivo de origen o destino y el tiempo que ha tardado en realizar esta operación. Antes de ejecutar estos comandos, y siempre que el usuario tenga permiso para ejecutarlos, Cassandra realiza una comprobación para verificar que el archivo existe y está en la ruta que el usuario le pasa por parámetro. Por último, el método *Show_doc* recibe la página web de la ayuda de elección de bases de datos.

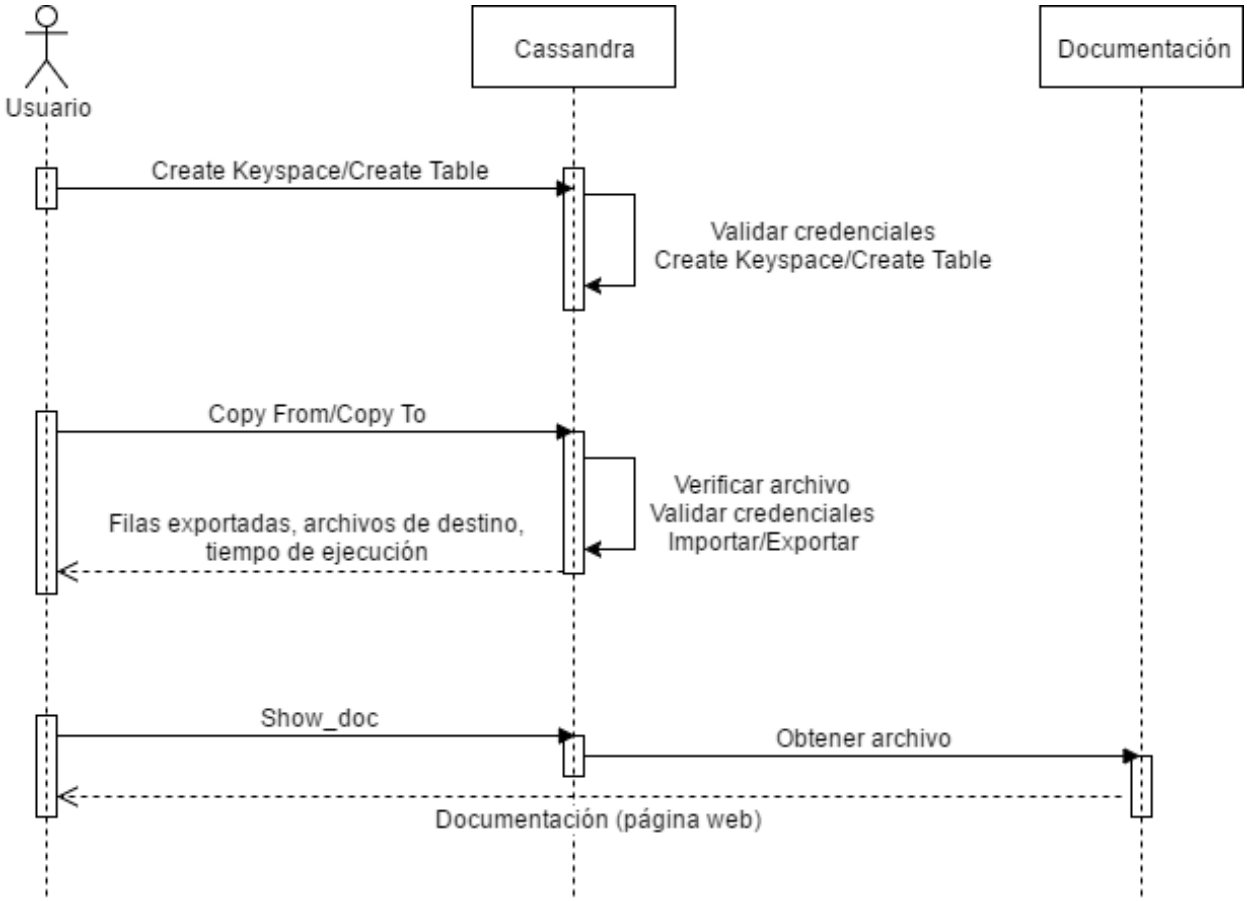

Ilustración 25: Diagrama de secuencia clase Cassandra

El siguiente representa los métodos contenidos en la clase Rol. Los métodos *Create*, *Alter* y *Drop* no reciben respuesta, y si el usuario tiene permiso para ejecutar esos comandos, Cassandra creará/modificará un rol con el nombre y propiedades que reciba del usuario, o borrará el rol que el usuario le indique. El método *List* recibe los roles que hay en ese momento creados en el sistema, y podrá ver qué rol tiene asignado el permiso de superusuario y el de login y cuál no, siempre y cuando posea los permisos para realizar esta petición.

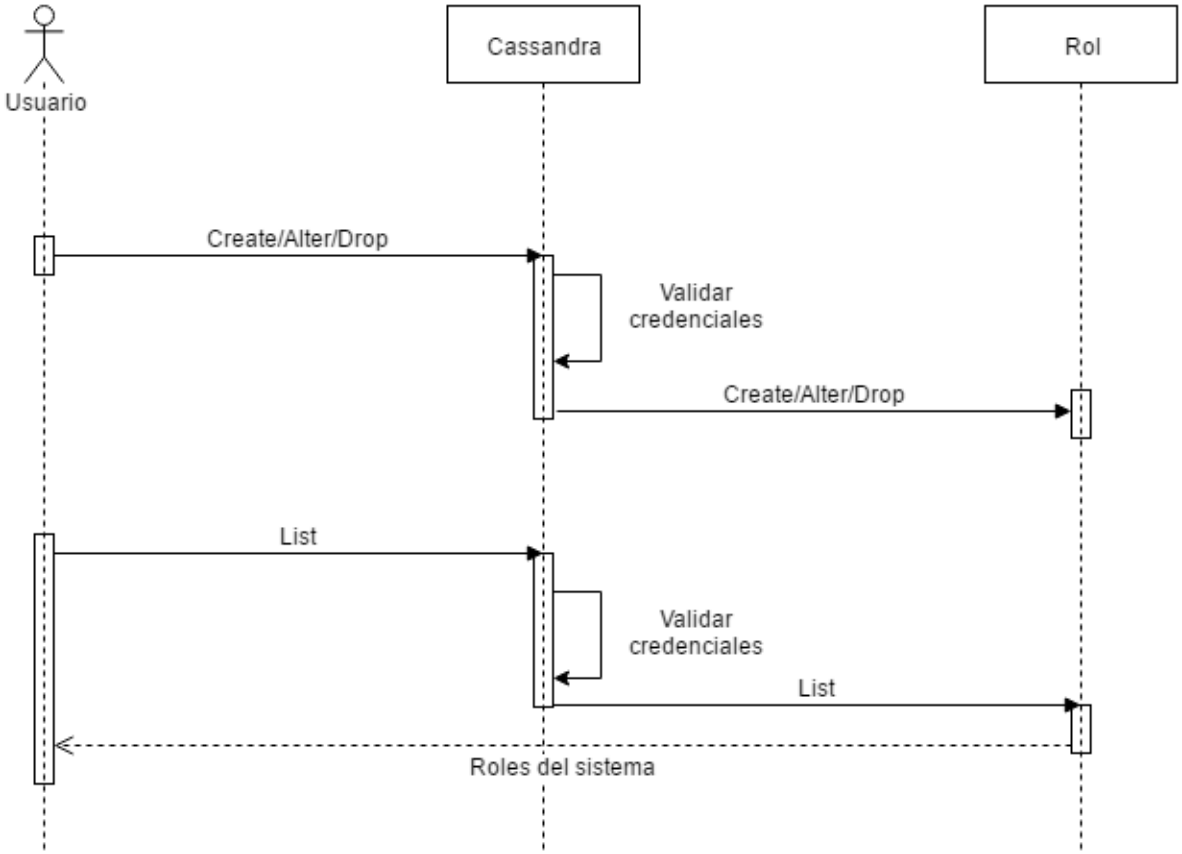

Ilustración 26: Diagrama de secuencia clase Rol

El siguiente representa los métodos contenidos en la clase Permiso. Los métodos *Grant* y *Revoke* no reciben respuesta, y si el usuario tiene permiso para ejecutar esta acción, Cassandra asignará o eliminará el permiso que reciba por parámetro del rol cuyo nombre también recibe del usuario. El método *List* recibe como respuesta los datos asociados al rol que el usuario pase por parámetro, siempre que éste tenga los permisos necesarios.

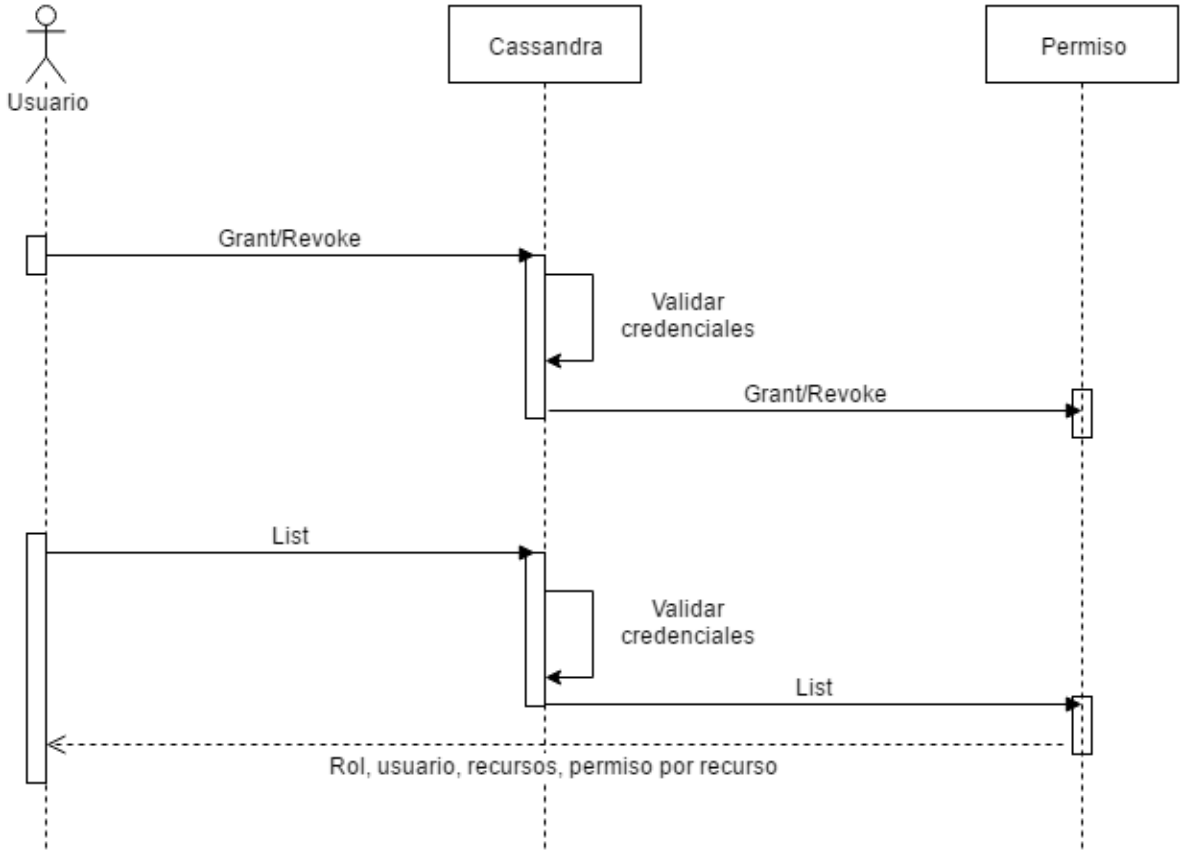

Ilustración 27: Diagrama de secuencia clase Permiso

Para terminar, el siguiente representa los métodos contenidos en la clase Usuarios. Los métodos *Insert*, *Update*, *Delete* y *Truncate* no reciben respuesta, y si el usuario tiene permiso para ejecutar esta acción, Cassandra realizará la opción solicitada con o sobre los datos que el usuario le pase en la petición. El método *Select* recibe los datos que el usuario haya pedido. Para pedirlos tendrá que tener permiso, y pasarle por parámetro a Cassandra alguna información del usuario para buscarle en la base de datos.

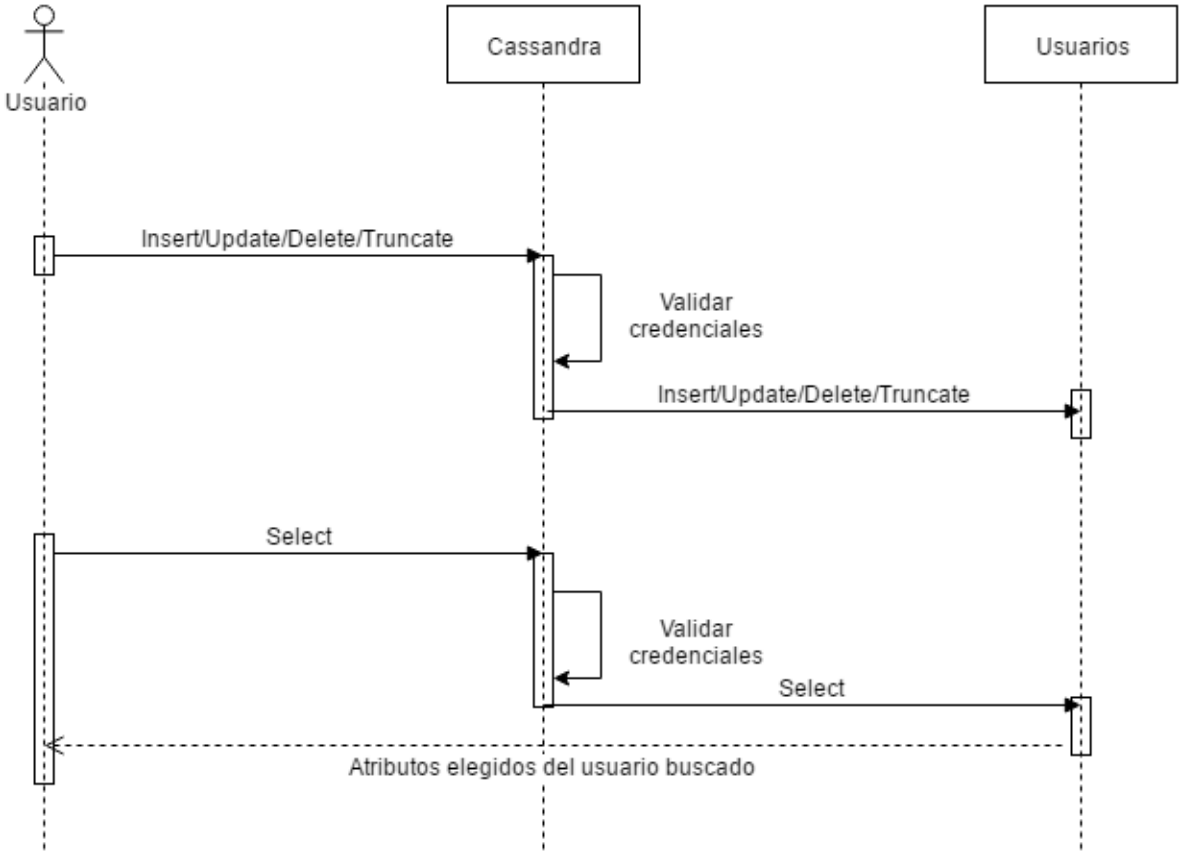

Ilustración 28: Diagrama de secuencia clase Usuarios

4.6 – Revisión de la interfaz de usuario

En este apartado se describe la interfaz de usuario que proporciona Cassandra. Al haber sido instalada en una distribución de Linux, la interfaz por la que interactuamos con Cassandra para realizar operaciones contra la base de datos es el terminal de Ubuntu. Esta interfaz se compone de una sola ventana donde se introduce el comando necesario para acceder a Cassandra, y, una vez dentro del sistema, ejecutar operaciones contra el mismo.

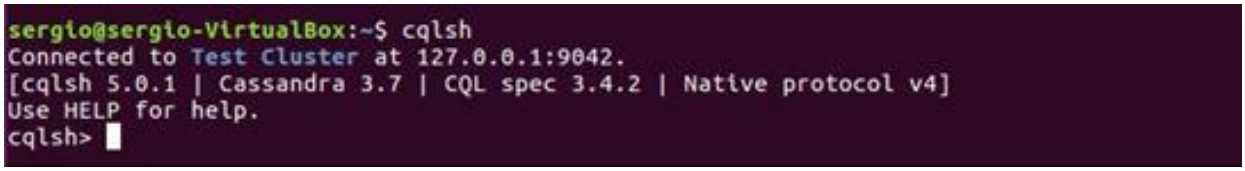

Ilustración 29: Interfaz gráfica de Cassandra

Sin embargo, al igual que muchas otras bases de datos, ya sean relacionales o no, Cassandra cuenta con un driver que permite que se acceda a ella desde el entorno de programación Eclipse. Aunque he realizado algunas pruebas para comprobar que, efectivamente, se podía conectar (*ILUSTRACIÓN 30: INTERFAZ GRÁFICA DE ECLIPSE*), he decidido seguir utilizando el terminal como interfaz de acceso a Cassandra, ya que me he familiarizado con esta interfaz y me resulta más sencilla de usar.

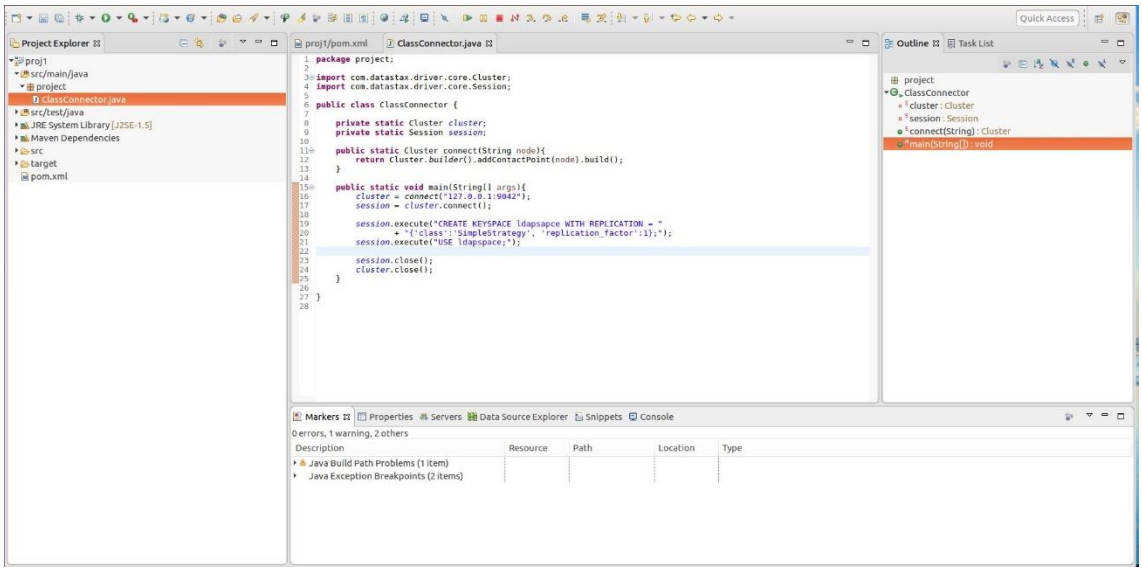

Ilustración 30: Interfaz gráfica de Eclipse

5 – Implementación

En este apartado se va a explicar de forma sencilla el funcionamiento del código desarrollado, necesario para realizar las tareas que se explicarán posteriormente en *6 – IMPLANTACIÓN*.

Para el desarrollo de dichas tareas he creado, y obtenido en algunos casos, los siguientes archivos:

- codigoHTML.txt
- codigo1.c
- salida.sh
- codigo2.c
- codigoPHP.php
- correos.txt
- codigo3.c
- datosLDAP.txt
- codigo3_toJSON.c
- datosLDAP_JSON.txt
- generated.json
- result.csv

5.1 – Diagrama de flujo

En este apartado se encuentra el diagrama que indica el flujo de obtención y uso de cada uno de los archivos mencionados anteriormente. El diagrama se ha desarrollado por medio de la herramienta Draw.io [26]. En él se puede seguir visualmente el proceso de obtención de los datos del Personal del Departamento de Informática, que quedarán almacenados finalmente en el archivo *datosLDAP.txt*.

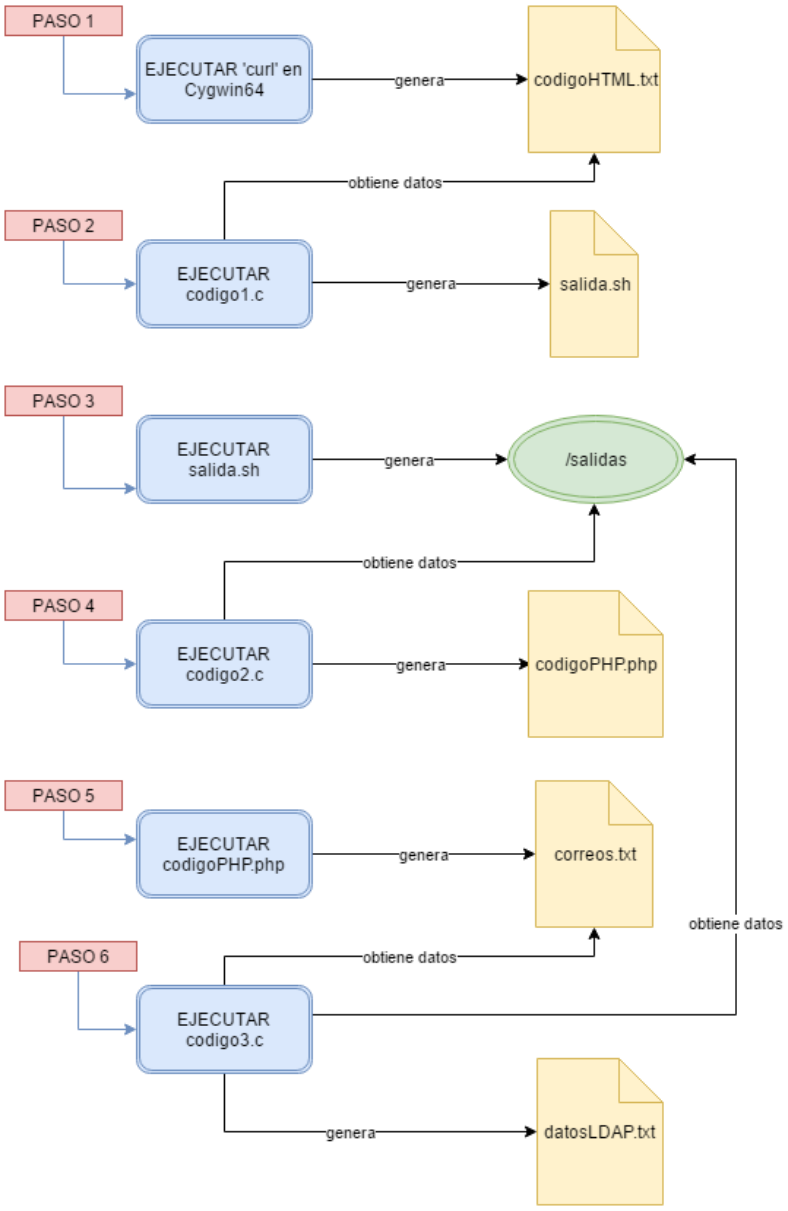

Ilustración 31: Diagrama de flujo de los códigos

5.2 – Archivos

La descripción detallada de cada uno de los archivos se encuentra en el Anexo *APÉNDICE IV - DESCRIPCIÓN DEL CÓDIGO*.

6 – Implantación

Este apartado describe el proceso de implantación del sistema, así como la preparación de las pruebas que se detallan en *7 – EVALUACIÓN* y una referencia al *APÉNDICE III – MANUAL DE INSTALACIÓN*.

A continuación, se detallan las distintas fases que componen el proceso de preparación anteriormente mencionado:

- **6.1 – Extracción de los datos**. Proceso de extracción de los datos contenidos en el LDAP para su posterior inserción en el sistema NoSQL elegido.
- **6.2 – Instalación de la solución.** Instalar en la máquina virtual la base de datos elegida anteriormente.
- **6.3 – Modo de uso.** Documentar el modo de uso y los comandos con los que cuenta el sistema elegido, algunos de los cuales serán usados para realizar las pruebas.

6.1 – Extracción de los datos

En primer lugar, me gustaría comentar que recuperar los datos mediante la descarga del código HTML no fue mi primera opción. Mi tutor me indicó que intentara ponerme en contacto con los administradores del LDAP para ver si tenían toda la información que necesito para el proyecto almacenada en un archivo que pudieran pasarme por correo. Así hice, y me puse en contacto con el Punto de Información del Campus de Leganés para transmitirles mi petición. Ellos redirigieron la misma al Servicio de Informática y Comunicaciones, los cuales, desafortunadamente, me comunicaron que no podían satisfacer mi petición, y que, si deseaba consultar la información de algún miembro de un departamento, utilizara el directorio de la universidad. El problema es que consultar a través del navegador la información de todas las personas del Departamento de Informática y copiarlas a un archivo es una tarea larga y poco o nada optimizada, por lo que la segunda opción para obtener los datos era a través de la descarga del código HTML de cada miembro del departamento y posterior extracción de la información almacenada en los archivos descargados.

Aprovechando los conocimientos adquiridos en la empresa en la que actualmente me encuentro trabajando desde septiembre, Everis Spain S.L.U., he utilizado la herramienta Cygwin64, con el comando *curl*, para la descarga del código HTML de la página web de la universidad. El código utilizado en el proceso se encuentra alojado en el repositorio [33] de GitHub.

El proceso que se ha seguido para lo obtención de los datos, el cual viene explicado gráficamente en *5.1 – DIAGRAMA DE FLUJO*, está detalladamente explicado en el Anexo *APÉNDICE V - OBTENCIÓN DE LOS DATOS*.

6.2 – Instalación de la solución

Una vez listo el programa VirtualBox (ver *INSTALACIÓN DE ORACLE VM VIRTUALBOX*), procedemos a crear la máquina virtual. Asignaremos a la máquina virtual 5 GB de RAM y 50 GB de disco duro, y como SO he elegido Ubuntu de 64 bits. Del mismo modo, he decidido clonar la máquina virtual y usar la clonada para realizar la implantación, dejando así la original intacta, para eliminar la máquina virtual dañada y clonarla de nuevo si surgiera algún problema. Este proceso se encuentra detalladamente explicado en *CREACIÓN DE LA MÁQUINA VIRTUAL*.

Una vez creada la máquina virtual, pasamos a instalar el sistema elegido. Este proceso se encuentra detalladamente explicado en *INSTALACIÓN DE CASSANDRA*.

6.3 – Modo de uso

En este apartado se explican los comandos y el modo de uso de la BBDD elegida.

6.3.1 – Cassandra

Una vez que tengamos instalado Cassandra en la máquina virtual (en caso contrario, ver *INSTALACIÓN DE CASSANDRA*), lo primero que necesitamos es abrir un terminal, en el que introducimos:

cqlsh

Lo siguiente es crear un espacio de trabajo, o *keyspace*, introduciendo en el terminal:

CREATE KEYSPACE keyspaceName WITH REPLICATION =

{'class':'replicationStrategy', 'replication_factor':replicationNumber};

Esto puede considerarse como una base de datos dentro de Cassandra. Podemos tener muchos keypsaces, pero se recomienda uno por aplicación. En *replicationStrategy* podemos poner *SimpleStrategy*, si sólo tenemos un centro de datos, o *NetworkTopologyStrategy*, si tenemos varios centros de datos; y en *replicationNumber* ponemos el número de nodos que contendrán copias de cada fila de datos. El nivel de consistencia depende de los valores que introduzcamos en estos dos campos al crear el *keyspace* [34]. Eso sí, a mayor consistencia mayor coste de despliegue. Para usar el espacio creado, introducimos en el terminal:

USE keyspaceName;

A partir de ahora, la sintaxis de Cassandra es prácticamente idéntica a la de un SGBD Relacional. Algunos de sus comandos son [35]:

- **ALTER KEYSPACE**: cambia el valor de las propiedades del *keyspace*.
- **CREATE TABLE**: define una nueva tabla.
- **DROP TABLE**: elimina una tabla.
- **INSERT**: añade o actualiza columnas.
- **DELETE**: elimina filas enteras, o una o más columnas de una o más filas.
- **SELECT**: devuelve los datos de una tabla.
- **UPDATE**: actualiza las columnas de una fila.

Sin embargo, el método *INSERT* no es útil para introducir muchos datos al mismo tiempo. Por suerte, Cassandra nos proporciona un método para introducir los datos desde un archivo externo.

Para importar un archivo en Cassandra necesitamos tener el documento en formato CSV. Para ello, creamos el archivo *codigo3_toJSON.c* para que devuelva los datos en un formato apto para la página web JSON Generator [36], y lo que nos devuelve lo guardamos en *generated.json*. Después cogemos los datos del archivo *generated.json* y los copiamos en [37], la cual nos muestra los datos en una tabla (*ILUSTRACIÓN 32: INTERFAZ DE LA WEB KONKLONE*). Después le damos a '*Download the entire CSV*', y se nos descarga un archivo llamado *result.csv*. Ahora ya podemos insertarlo en Cassandra.

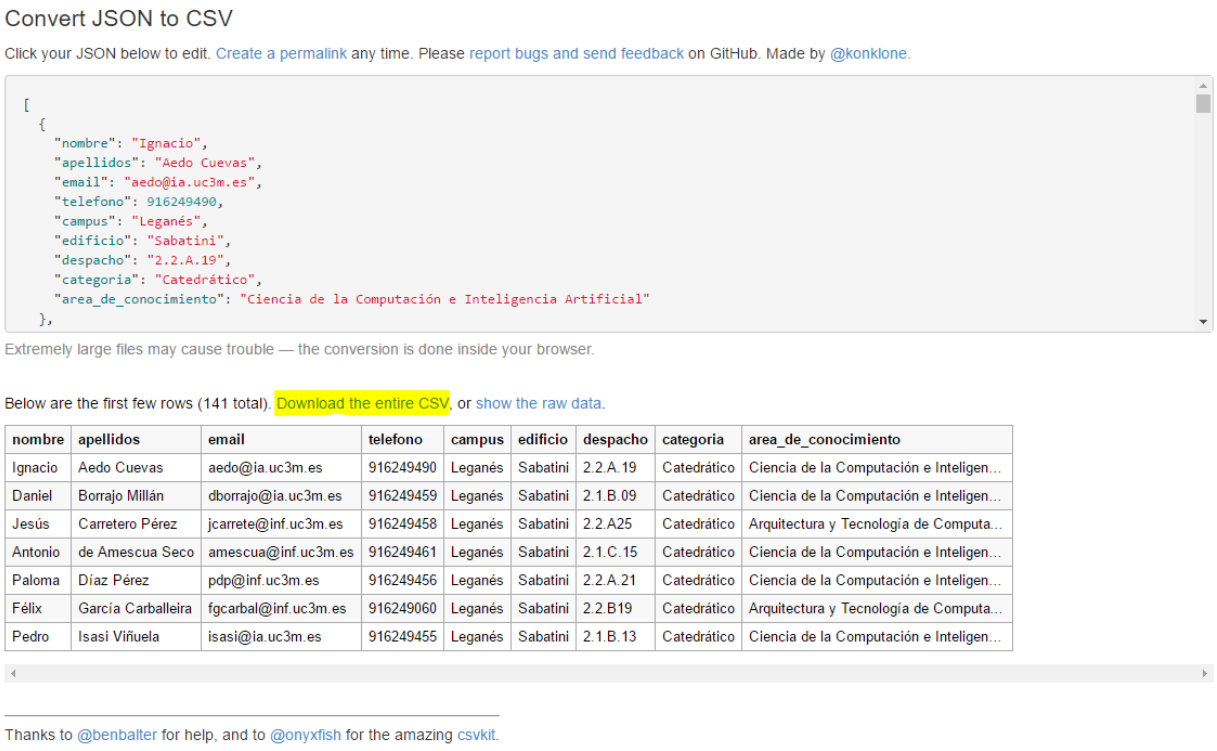

Ilustración 32: Interfaz de la web Konklone

Para insertar el archivo en Cassandra, introducimos en el terminal:

COPY tableName (atributo1, atributo2, …) FROM 'fileName.csv';

Si el archivo no está en el mismo directorio que Cassandra, hay que introducir la ruta en la que se encuentra el archivo (por ejemplo, /home/user/Escritorio/file.csv). Del mismo modo, si queremos exportar la información de la base de datos a un archivo externo, introducimos en el terminal:

COPY tableName (atributo1, atributo2, …) TO 'fileName.csv';

En el Anexo *APÉNDICE VI - EJEMPLOS PRÁCTICOS DE CASSANDRA* se exponen algunos ejemplos prácticos para demostrar que Cassandra cumple con las características indicadas en *3.3 – CARACTERÍSTICAS DE LA SOLUCIÓN DESEADA*.

7 – Evaluación

Este apartado describe todo lo relacionado con el plan de pruebas del proyecto. Este plan nos sirve para garantizar que todas las características que tenemos del sistema, que han quedado recogidas en *3.5 – REQUISITOS*, se cumplen sin ningún tipo de excepción, asegurando así el buen funcionamiento del mismo.

7.1 – Especificación del plan de pruebas

Este plan de pruebas va a estar compuesto por cuatro tipos de pruebas fundamentales:

- **Pruebas de validación**. Su objetivo es comprobar que el sistema cumple con las funciones prometidas.
- **Pruebas de restricción**. Su objetivo es verificar que se cumplen las restricciones del sistema.
- **Pruebas de compatibilidad**. Su objetivo es comprobar que el sistema funciona en diferentes entornos.
- **Pruebas de rendimiento**. Su objetivo es determinar lo rápido que un sistema realiza una tarea en ciertas condiciones de trabajo.

7.2 – Especificación técnica del plan de pruebas

En este apartado se definen las pruebas realizadas. Para documentar estas pruebas, cada una de ellas vendrá definida en una tabla como la siguiente:

Identificador	
Objetivo	
Necesidades del entorno	
Clase	
Método	
Entrada	
Secuencia	
Salida	
Requisitos relacionados	

Tabla 81: Plantilla para pruebas

Cada prueba contendrá los campos a continuación descritos:

- **Identificador**. Cada prueba tendrá un identificador propio del tipo PX-YY para diferenciarlo de los demás, donde:
  - **P**. Indica que es una prueba.
  - **X**. Indica el tipo de prueba, que puede ser:
    - **V**. Prueba de validación
    - **R**. Prueba de restricción.
    - **C**. Prueba de compatibilidad.
    - **E**. Prueba de rendimiento.
  - **YY**. Indica el número de prueba, que va de 01 a 99.
- **Objetivo**. El propósito de la prueba.
- **Necesidades del entorno**. Condiciones necesarias para llevar a cabo la prueba.
- **Clase**. Clase a la que pertenece el método que se va a probar (sólo aplica para las pruebas de validación).
- **Método**. Nombre del método que se va a probar (sólo aplica para las pruebas de validación).
- **Entrada**. Valores de entrada necesarios para realizar la prueba.
- **Secuencia**. Pasos necesarios para llevar a cabo la prueba.
- **Salida**. Valores de salida tras la realización de la prueba.
- **Requisitos relacionados**. Requisitos de usuario que quedan verificados tras la prueba (sólo aplica para las pruebas de validación y restricción).

7.2.1 – Pruebas de validación

En este apartado se encuentran las pruebas de validación, que permiten verificar el correcto funcionamiento del sistema.

PV-01	
Objetivo	Comprobar que el sistema permite la inserción de un usuario.
Necesidades del entorno	El *keyspace* y la tabla donde se va a insertar el usuario deben estar creadas y se debe tener permiso para ejecutar la inserción.
Clase	Usuario
Método	insert
Entrada	*Keyspace* y tabla donde se va a insertar, atributos de la tabla y datos para esos atributos.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar una inserción. Si lo tiene, inserta los datos en la tabla; si no lo tiene, no se ejecuta la operación.
Salida	Ninguna o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-01, RSF-01

Tabla 82: Prueba de validación PV-01

PV-02	
Objetivo	Comprobar que el sistema permite la inserción de datos desde un archivo.
Necesidades del entorno	El *keyspace* y la tabla donde se va a insertar el usuario deben estar creadas, se debe tener permiso para ejecutar la inserción y el archivo con los datos debe existir.
Clase	Cassandra
Método	copy from
Entrada	*Keyspace* y tabla donde se va a insertar y ruta donde se encuentra el archivo.
Secuencia	Cassandra comprueba primero que el archivo exista. Si es así, comprueba si el usuario tiene permiso para realizar una inserción. Si lo tiene, inserta los datos en la tabla; si no lo tiene, o Cassandra no encuentra el archivo, no se ejecuta la operación.
Salida	Ninguna, un mensaje diciendo que no tiene permiso para ejecutar esa acción o un mensaje diciendo que no se ha encontrado el archivo.
Requisitos relacionados	RUC-01, RSF-02

Tabla 83: Prueba de validación PV-02

PV-03	
Objetivo	Comprobar que el sistema permite la modificación de un usuario.
Necesidades del entorno	El *keyspace*, la tabla y el usuario deben estar creados y se debe tener permiso para ejecutar la actualización.
Clase	Usuario
Método	update
Entrada	*Keyspace* y tabla donde está el usuario a modificar, atributos de la tabla que se van a cambiar, datos para esos atributos y clave primaria del usuario, para que Cassandra lo encuentre.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar una actualización. Si lo tiene, busca al usuario y modifica sus datos; si no lo tiene, no se ejecuta la operación.
Salida	Ninguna o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-01, RSF-03

Tabla 84: Prueba de validación PV-03

PV-04	
Objetivo	Comprobar que el sistema permite la modificación de datos desde un archivo.
Necesidades del entorno	El *keyspace*, la tabla y el usuario deben estar creados, se debe tener permiso para ejecutar la actualización y el archivo con los datos debe existir.
Clase	Cassandra
Método	copy from
Entrada	*Keyspace* y tabla donde está el usuario a modificar y ruta donde se encuentra el archivo.
Secuencia	Cassandra comprueba primero que el archivo exista. Si es así, comprueba si el usuario tiene permiso para realizar una actualización. Si lo tiene, modifica los datos de la tabla; si no lo tiene, o Cassandra no encuentra el archivo, no se ejecuta la operación.
Salida	Ninguna, un mensaje diciendo que no tiene permiso para ejecutar esa acción o un mensaje diciendo que no se ha encontrado el archivo.
Requisitos relacionados	RUC-01, RSF-04

Tabla 85: Prueba de validación PV-04

PV-05	
Objetivo	Comprobar que el sistema permite el borrado de un usuario.
Necesidades del entorno	El *keyspace*, la tabla y el usuario deben estar creados y se debe tener permiso para ejecutar el borrado.
Clase	Usuario
Método	delete
Entrada	*Keyspace* y tabla donde está el usuario a borrar, atributos de la tabla que se quieren borrar (si no se introduce, se borra la fila entera) y clave primaria del usuario, para que Cassandra lo encuentre.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar una un borrado. Si lo tiene, busca al usuario y borra sus datos; si no lo tiene, no se ejecuta la operación.
Salida	Ninguna o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-01, RSF-05

Tabla 86: Prueba de validación PV-05

PV-06	
Objetivo	Comprobar que el sistema permite el borrado de todos los datos.
Necesidades del entorno	El *keyspace* y la tabla deben estar creadas y se debe tener permiso para ejecutar el borrado.
Clase	Usuario
Método	truncate
Entrada	*Keyspace* y tabla que se quiere borrar.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar un borrado. Si lo tiene, borra los datos de la tabla; si no lo tiene, no se ejecuta la operación.
Salida	Ninguna o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-01, RSF-06

Tabla 87: Prueba de validación PV-06

PV-07	
Objetivo	Comprobar que el sistema permite buscar usuarios por cualquier atributo.
Necesidades del entorno	El *keyspace*, la tabla y el usuario deben estar creados y se debe tener permiso para ejecutar la búsqueda.
Clase	Usuario
Método	select
Entrada	*Keyspace* y tabla donde se quiere buscar y datos del usuario para buscarlo.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar una consulta. Si lo tiene, busca en la tabla y devuelve la información; si no lo tiene, no se ejecuta la operación.
Salida	Aquellos usuarios que contengan los datos introducidos para la búsqueda o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-02, RSF-07

Tabla 88: Prueba de validación PV-07

PV-08	
Objetivo	Comprobar que el sistema permite buscar usuarios por cualquier atributo.
Necesidades del entorno	El *keyspace*, la tabla y el usuario deben estar creados y se debe tener permiso para ejecutar la búsqueda.
Clase	Usuario
Método	select
Entrada	*Keyspace* y tabla donde se quiere buscar y otros datos distintos del usuario para buscarlo.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar una consulta. Si lo tiene, busca en la tabla y devuelve la información; si no lo tiene, no se ejecuta la operación.
Salida	Aquellos usuarios que contengan los datos introducidos para la búsqueda o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-02, RSF-07

Tabla 89: Prueba de validación PV-08

PV-09	
Objetivo	Comprobar que el sistema permite exportar los datos a un archivo.
Necesidades del entorno	El k*eyspace* y la tabla con los datos deben estar creadas, se debe tener permiso para ejecutar la exportación y el archivo de destino debe existir.
Clase	Cassandra
Método	copy to
Entrada	*Keyspace* y tabla de donde se van a sacar los datos y ruta donde se encuentra el archivo de destino.
Secuencia	Cassandra comprueba primero que el archivo exista. Si es así, comprueba si el usuario tiene permiso para realizar una exportación. Si lo tiene, inserta los datos de la tabla en el archivo; si no lo tiene, o Cassandra no encuentra el archivo, no se ejecuta la operación.
Salida	Ninguna, un mensaje diciendo que no tiene permiso para ejecutar esa acción o un mensaje diciendo que no se ha encontrado el archivo.
Requisitos relacionados	RUC-03, RSF-08

Tabla 90: Prueba de validación PV-09

PV-10	
Objetivo	Comprobar que el usuario cuenta con una ayuda para la elección de una base de datos NoSQL.
Necesidades del entorno	La página web con la documentación debe estar creada.
Clase	Cassandra
Método	show_doc
Entrada	Ninguna.
Secuencia	El contenido del apartado *2.3.4 – RESUMEN* se encontraría en una página web a la que se accede a través de Cassandra. El usuario que desea consultar esta documentación ejecuta el método *show_doc* y Cassandra le devuelve el archivo HTML con esta información. Esta funcionalidad no se ha podido probar porque no se encuentra implementada, por lo que se ha dejado como trabajo futuro.
Salida	La página web con la documentación de ayuda para la elección de una base de datos.
Requisitos relacionados	RUC-04, RSF-09

Tabla 91: Prueba de validación PV-10

PV-11	
Objetivo	Comprobar que el sistema permite crear un rol.
Necesidades del entorno	Se debe tener permiso para ejecutar la operación.
Clase	Rol
Método	create
Entrada	Nombre de usuario y contraseña que no existan ya y, opcionalmente, valor de los permisos de superusuario y login.
Secuencia	Cassandra comprueba primero si el usuario tiene permiso para realizar la operación. Si es así, comprueba si ese nombre de usuario y contraseña existe ya. Si no existe, crea el rol; si existe, no se ejecuta la operación.
Salida	Ninguna o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-05, RSF-10

Tabla 92: Prueba de validación PV-11

PV-12	
Objetivo	Comprobar que el sistema permite modificar un rol.
Necesidades del entorno	Se debe tener permiso para ejecutar la operación y el rol debe existir.
Clase	Rol
Método	alter
Entrada	Nombre de usuario y contraseña del rol a modificar y valor de los permisos de superusuario y login.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar la operación. Si lo tiene, modifica el rol; si no lo tiene, no se ejecuta la operación.
Salida	Ninguna o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-05, RSF-11

Tabla 93: Prueba de validación PV-12

PV-13	
Objetivo	Comprobar que el sistema permite borrar un rol.
Necesidades del entorno	Se debe tener permiso para ejecutar la operación y el rol debe existir.
Clase	Rol
Método	drop
Entrada	Nombre de usuario.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar la operación. Si lo tiene y el rol a borrar no es el rol que realiza la petición, borra el rol; si no, no se ejecuta la operación.
Salida	Ninguna, un mensaje diciendo que no tiene permiso para ejecutar esa acción o un mensaje diciendo que el rol a borrar es el que se está usando.
Requisitos relacionados	RUC-05, RSF-12

Tabla 94: Prueba de validación PV-13

PV-14	
Objetivo	Comprobar que el sistema permite asignar permisos a un rol.
Necesidades del entorno	Se debe tener permiso para ejecutar la operación y el rol al que se le van a asignar los permisos debe existir, así como el recurso sobre el que se le van a dar los permisos.
Clase	Permiso
Método	grant
Entrada	Permiso a conceder, recurso sobre el que se le concede (*keyspace* o tabla) y nombre de usuario al que se le concede.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar la operación. Si lo tiene, concede el permiso; si no lo tiene, no se ejecuta la operación.
Salida	Ninguna o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-06, RSF-13

Tabla 95: Prueba de validación PV-14

PV-15	
Objetivo	Comprobar que el sistema permite eliminar los permisos de un rol.
Necesidades del entorno	Se debe tener permiso para ejecutar la operación y el rol al que se le van a revocar los permisos debe existir, así como el recurso sobre el que se van a quitar los permisos.
Clase	Permiso
Método	revoke
Entrada	Permiso a revocar, recurso sobre el que se le revoca (*keyspace* o tabla) y nombre de usuario al que se le elimina el permiso.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar la operación. Si lo tiene, revoca el permiso; si no lo tiene, no se ejecuta la operación.
Salida	Ninguna o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-06, RSF-14

Tabla 96: Prueba de validación PV-15

PV-16	
Objetivo	Comprobar que el sistema permite listar los roles existentes.
Necesidades del entorno	Se debe tener permiso para ejecutar la operación.
Clase	Rol
Método	list
Entrada	Ninguna.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar la operación. Si lo tiene, lista los roles existentes; si no lo tiene, no se ejecuta la operación.
Salida	Lista de roles del sistema o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-07, RSF-15

Tabla 97: Prueba de validación PV-16

PV-17	
Objetivo	Comprobar que el sistema permite listar los permisos de un rol.
Necesidades del entorno	Se debe tener permiso para ejecutar la operación y el rol debe existir.
Clase	Permiso
Método	list
Entrada	Rol del que se quieren ver los permisos.
Secuencia	Cassandra comprueba si el usuario tiene permiso para realizar la operación. Si lo tiene, lista los permisos; si no lo tiene, no se ejecuta la operación.
Salida	Lista de permisos del rol elegido o un mensaje diciendo que no tiene permiso para ejecutar esa acción.
Requisitos relacionados	RUC-07, RSF-16

Tabla 98: Prueba de validación PV-17

PV-18	
Objetivo	Comprobar que sólo es necesario realizar 3 comandos para inicializar el sistema.
Necesidades del entorno	El archivo con los datos debe existir.
Clase	Cassandra
Método	create keyspace, create table, copy from
Entrada	Nombre del *keyspace*, nombre y atributos de la tabla y ruta donde se encuentra el archivo con los datos.
Secuencia	Primero se ejecuta el método *create keyspace* para crear el mismo, luego se ejecuta el método *create table* para crear la tabla, y por último se ejecuta el método *copy from* para cargar en la tabla recién creada los datos del archivo. De esta forma, con sólo 3 comandos tenemos el sistema inicializado y listo para ser usado.
Salida	Ninguna o un mensaje diciendo que no se ha encontrado el archivo.
Requisitos relacionados	RUC-08, RSF-17

Tabla 99: Prueba de validación PV-18

7.2.2 – Pruebas de restricción

En este apartado se encuentran las pruebas de restricción, que permiten verificar el cumplimiento las restricciones del sistema.

PR-01	
Objetivo	Comprobar que el sistema sólo permite realizar inserciones, modificaciones y borrados al administrador.
Necesidades del entorno	La autenticación y autorización deben estar habilitadas en el archivo de configuración y el rol del administrador debe estar creado y con los permisos necesarios.
Entrada	Nombre de usuario y contraseña del administrador y de otro usuario no administrador, y un *keyspace* y tabla creadas y con datos.
Secuencia	Se listan los permisos del rol administrador para comprobar que, en teoría, tiene permiso de escritura sobre la base de datos, y se listan los permisos del rol no administrador para comprobar que no tiene permiso de escritura. Para demostrar esto, se inicia sesión con el rol del administrador, se ejecuta una operación de escritura sobre la base de datos, y se comprueba que, efectivamente, puede realizar escrituras. Después se inicia sesión con el rol no administrador y se hace lo mismo, para comprobar que este rol no puede realizar escrituras.
Salida	Sólo el usuario con el rol de administrador puede realizar escrituras sobre la base de datos.
Requisitos relacionados	RUR-01, RSN-07, RSN-08, RSN-09

Tabla 100: Prueba de restricción PR-01

PR-02	
Objetivo	Comprobar que el sistema permite realizar consultas a cualquier usuario.
Necesidades del entorno	La autenticación y autorización deben estar habilitadas en el archivo de configuración y los roles a probar deben estar creados y con los permisos necesarios.
Entrada	Nombre de usuario y contraseña de los roles a probar, y un *keyspace* y tabla creadas y con datos.
Secuencia	Se listan los permisos de los roles a probar para comprobar que, en teoría, tienen permiso de lectura sobre la base de datos. Para demostrar esto, se inicia sesión con cualquier rol existente en el sistema, se ejecuta una operación de lectura sobre la base de datos, y se comprueba que, efectivamente, puede realizar lecturas.
Salida	Cualquier usuario puede realizar una lectura sobre la base de datos, pues todos los roles del sistema deben tener, como mínimo, permiso de login y lectura.
Requisitos relacionados	RUR-01, RSN-06

Tabla 101: Prueba de restricción PR-02

PR-03	
Objetivo	Comprobar que el sistema no tarda más de 5 minutos en restablecerse tras un fallo de software.
Necesidades del entorno	El *keyspace* debe estar creado con un factor de replicación mínimo de 2, o tener los datos actualizados en un archivo.
Entrada	Ninguna.
Secuencia	Cuando haya un fallo de software, se reinicia el sistema por medio del comando *service cassandra restart* y se cargan los datos del archivo en el caso de que se haya perdido la información.
Salida	El sistema se reinicia y vuelve a estar disponible en menos de 5 minutos.
Requisitos relacionados	RUR-02, RSN-10

Tabla 102: Prueba de restricción PR-03

PR-04	
Objetivo	Comprobar que el sistema no tarda más de 5 segundos en devolver un dato.
Necesidades del entorno	Se debe tener un *keyspace* y una tabla creadas y con datos.
Entrada	*Keyspace* y tabla donde se quiere buscar y datos del usuario para buscarlo.
Secuencia	Se guarda en un archivo la sentencia a ejecutar y se utiliza la opción -f de la consola cql de Cassandra para ejecutar la sentencia de dicho archivo. Al mismo tiempo, se controla el tiempo que tarda en ejecutar la sentencia usando el comando *time*.
Salida	El dato es devuelto en menos de 5 segundos.
Requisitos relacionados	RUR-03, RSN-11

Tabla 103: Prueba de restricción PR-04

PR-05	
Objetivo	Comprobar que el sistema es accesible desde otra máquina.
Necesidades del entorno	La máquina local debe estar encendida y funcionando.
Entrada	Dirección IP y puerto de la máquina donde esté el sistema.
Secuencia	Se introduce la dirección IP y puerto de la máquina local en alguna herramienta de gestión de acceso remoto. Como para este proyecto la implantación se ha llevado a cabo en una máquina virtual, se ha utilizado la herramienta de acceso remoto de Windows, usando la dirección IP de la máquina local y el puerto especificado para VirtualBox [38].
Salida	Poder ejecutar operaciones sobre Cassandra desde un equipo remoto.
Requisitos relacionados	RUR-04, RSN-12

Tabla 104: Prueba de restricción PR-05

PR-06	
Objetivo	Comprobar que el sistema de ayuda cuenta con una documentación.
Necesidades del entorno	La página web de la documentación debe estar creada.
Entrada	Ninguna.
Secuencia	La página web debe contar con la información de los apartados *2.1 – INTRODUCCIÓN* y *2.3 – COMPARATIVA* para facilitar el entendimiento de las distintas opciones que proporciona el árbol de decisión. Esta funcionalidad no se ha podido probar porque no se encuentra implementada, por lo que se ha dejado como trabajo futuro.
Salida	Página web con la información de dichos apartados.
Requisitos relacionados	RUR-05, RSN-13, RSN-14

Tabla 105: Prueba de restricción PR-06

PR-07	
Objetivo	Comprobar que sólo existe un rol superusuario.
Necesidades del entorno	La autenticación y autorización deben estar habilitadas en el archivo de configuración, debe existir al menos un rol y se debe tener permiso para listar los roles.
Entrada	Ninguna.
Secuencia	Con un rol que tenga el permiso para hacerlo, se ejecuta la operación de listar los roles.
Salida	Al listar los roles se comprueba que sólo uno de ellos tiene un valor *true* en el campo superusuario.
Requisitos relacionados	RUR-06, RSN-15

Tabla 106: Prueba de restricción PR-07

PR-08	
Objetivo	Comprobar que es necesario iniciar sesión para acceder al sistema.
Necesidades del entorno	La autenticación y autorización deben estar habilitadas en el archivo de configuración.
Entrada	Nombre de usuario y contraseña válidos.
Secuencia	El sistema compara el nombre de usuario y contraseña introducidos con los datos almacenados en los archivos de configuración. Si coinciden, se permite el acceso; si no coinciden, no se permite el acceso.
Salida	Acceso al sistema o mensaje de error informando que ese usuario y contraseña no tiene permiso de acceso.
Requisitos relacionados	RUR-07, RSN-16

Tabla 107: Prueba de restricción PR-08

PR-09	
Objetivo	Comprobar que el sistema sólo permite crear, modificar y borrar roles al administrador.
Necesidades del entorno	La autenticación y autorización deben estar habilitadas en el archivo de configuración y el rol del administrador debe estar creado y con los permisos necesarios.
Entrada	Nombre de usuario y contraseña del administrador y de otro usuario no administrador.
Secuencia	Se listan los permisos del rol administrador para comprobar que, en teoría, tiene permiso de gestión de roles, y se listan los permisos del rol no administrador para comprobar que no tiene permiso de gestión de roles. Para demostrar esto, se inicia sesión con el rol del administrador, se ejecuta una operación de gestión de roles, y se comprueba que, efectivamente, puede gestionar los roles. Después se inicia sesión con el rol no administrador y se hace lo mismo, para comprobar que este rol no puede gestionar los roles.
Salida	Sólo el usuario con el rol de administrador puede gestionar los roles.
Requisitos relacionados	RUR-08, RSN-17, RSN-18, RSN-19

Tabla 108: Prueba de restricción PR-09

PR-10	
Objetivo	Comprobar que el sistema sólo permite asignar y eliminar permisos al administrador.
Necesidades del entorno	La autenticación y autorización deben estar habilitadas en el archivo de configuración y el rol del administrador debe estar creado y con los permisos necesarios.
Entrada	Nombre de usuario y contraseña del administrador y de otro usuario no administrador.
Secuencia	Se listan los permisos del rol administrador para comprobar que, en teoría, puede gestionar los permisos, y se listan los permisos del rol no administrador para comprobar que no puede gestionar los permisos. Para demostrar esto, se inicia sesión con el rol del administrador, se ejecuta una operación de gestión de permisos, y se comprueba que, efectivamente, puede gestionar los permisos. Después se inicia sesión con el rol no administrador y se hace lo mismo, para comprobar que este rol no puede gestionar los permisos.
Salida	Sólo el usuario con el rol de administrador puede gestionar los permisos.
Requisitos relacionados	RUR-09, RSN-20, RSN-21

Tabla 109: Prueba de restricción PR-10

PR-11	
Objetivo	Comprobar que el sistema sólo permite listar roles y permisos al administrador.
Necesidades del entorno	La autenticación y autorización deben estar habilitadas en el archivo de configuración y el rol del administrador debe estar creado y con los permisos necesarios.
Entrada	Nombre de usuario y contraseña del administrador y de otro usuario no administrador.
Secuencia	Se listan los permisos del rol administrador para comprobar que, en teoría, tiene permiso de listado, y se listan los permisos del rol no administrador para comprobar que no tiene permiso de listado. Para demostrar esto, se inicia sesión con el rol del administrador, se ejecuta una operación de listado, y se comprueba que, efectivamente, puede realizarla. Después se inicia sesión con el rol no administrador y se hace lo mismo, para comprobar que este rol no puede listar los roles y permisos.
Salida	Sólo el usuario con el rol de administrador puede listar los roles y permisos.
Requisitos relacionados	RUR-10, RSN-22, RSN-23

Tabla 110: Prueba de restricción PR-11

7.2.3 – Pruebas de compatibilidad

En este apartado se encuentran las pruebas de compatibilidad, que permiten verificar el cumplimiento la portabilidad del sistema.

PC-01	
Objetivo	Comprobar que el sistema funciona en distintos sistemas operativos.
Necesidades del entorno	Se debe tener instalada la última versión de Java.
Entrada	Ninguna.
Secuencia	Como nos indican en la documentación de Cassandra, su instalación puede llevarse a cabo tanto en sistemas Linux como en Windows [39].
Salida	Cassandra puede ser instalada en varios sistemas operativos diferentes.

Tabla 111: Prueba de compatibilidad PC-01

7.2.4 – Pruebas de rendimiento

En este apartado se encuentran las pruebas de rendimiento, que permiten verificar la velocidad de ejecución del sistema. Únicamente se han medido las operaciones de inserción, actualización, borrado y consulta, pues serán las más frecuentes, en especial la de consulta. Para medir el tiempo de cada operación se ha utilizado el comando *time* de Unix, que calcula el tiempo de ejecución de la operación que se especifique después del comando.

Las especificaciones técnicas del equipo donde se han llevado a cabo estas pruebas de rendimiento están descritas en *3.7 – ENTORNO OPERACIONAL*.

PE-01	
Objetivo	Comprobar el rendimiento del sistema en una inserción.
Necesidades del entorno	El *keyspace* y la tabla deben estar creadas y con datos.
Entrada	10 nuevos usuarios a insertar.
Secuencia	Se han insertado 10 usuarios y se ha medido el tiempo que ha tardado cada inserción. Después, se han calculado los siguientes datos: - Media: **0.6037** - Varianza: **0.3792** - Desviación Típica: **0.6158** - Intervalo de confianza: **0.6037 ± 0.3816**
Salida	Viendo los resultados que salen, se puede afirmar que el sistema no tarda más de un 1 segundo en insertar un dato.

Tabla 112: Prueba de rendimiento PE-01

PE-02	
Objetivo	Comprobar el rendimiento del sistema en una actualización.
Necesidades del entorno	El *keyspace* y la tabla deben estar creadas y con datos y el usuario a modificar debe existir.
Entrada	10 usuarios a modificar.
Secuencia	Se han modificado 10 usuarios y se ha medido el tiempo que ha tardado cada actualización. Después, se han calculado los siguientes datos: - Media: **0.5217** - Varianza: **0.2761** - Desviación Típica: **0.5255** - Intervalo de confianza: **0.5217 ± 0.3257**
Salida	Viendo los resultados que salen, se puede afirmar que el sistema no tarda más de un 1 segundo en actualizar un dato.

Tabla 113: Prueba de rendimiento PE-02

PE-03	
Objetivo	Comprobar el rendimiento del sistema en un borrado.
Necesidades del entorno	El *keyspace* y la tabla deben estar creadas y con datos y el usuario a borrar debe existir.
Entrada	10 usuarios a borrar.
Secuencia	Se han borrado 10 usuarios y se ha medido el tiempo que ha tardado cada borrado. Después, se han calculado los siguientes datos: - Media: **0.528** - Varianza: **0.2796** - Desviación Típica: **0.5288** - Intervalo de confianza: **0.528 ± 0.3277**
Salida	Viendo los resultados que salen, se puede afirmar que el sistema no tarda más de un 1 segundo en borrar un dato.

Tabla 114: Prueba de rendimiento PE-03

PE-04	
Objetivo	Comprobar el rendimiento del sistema en una búsqueda.
Necesidades del entorno	El *keyspace* y la tabla deben estar creadas y con datos y el usuario a buscar debe existir.
Entrada	10 usuarios a consultar.
Secuencia	Se han buscado 10 usuarios y se ha medido el tiempo que ha tardado cada consulta. Después, se han calculado los siguientes datos: - Media: **0.5839** - Varianza: **0.3488** - Desviación Típica: **0.5906** - Intervalo de confianza: **0.5839 ± 0.366**
Salida	Viendo los resultados que salen, se puede afirmar que el sistema no tarda más de un 1 segundo en recuperar un dato.

Tabla 115: Prueba de rendimiento PE-04

7.3 – Matrices de trazabilidad

Una matriz de trazabilidad sirve para verificar requisitos, ya sea respecto a su procedencia, respecto a la funcionalidad del sistema que garantizan, o respecto a las pruebas realizadas.

7.3.1 – Matriz de trazabilidad entre requisitos de usuario y pruebas

A continuación se encuentra la matriz de trazabilidad, que verifica que todos los requisitos de usuario cuentan con una prueba.

		REQUISITOS DE USUARIO																	
		RUC-01	RUC-02	RUC-03	RUC-04	RUC-05	RUC-06	RUC-07	RUC-08	RUR-01	RUR-02	RUR-03	RUR-04	RUR-05	RUR-06	RUR-07	RUR-08	RUR-09	RUR-10
PRUEBAS	PV-01	✓																	
	PV-02	✓																	
	PV-03	✓																	
	PV-04	✓																	
	PV-05	✓																	
	PV-06	✓																	
	PV-07		✓																
	PV-08		✓																
	PV-09			✓															
	PV-10				✓														
	PV-11					✓													
	PV-12					✓													
	PV-13					✓													
	PV-14						✓												
	PV-15						✓												
	PV-16							✓											
	PV-17							✓											
	PV-18								✓										
	PR-01									✓									
	PR-02									✓									
	PR-03										✓								
	PR-04											✓							
	PR-05												✓						
	PR-06													✓					
	PR-07														✓				
	PR-08															✓			
	PR-09																✓		
	PR-10																	✓	
	PR-11																		✓

Tabla 116: Matriz de trazabilidad entre requisitos de usuario y pruebas

7.4 – Análisis de resultados

En este apartado vamos a comprobar cuáles de las pruebas anteriormente definidas han dado el resultado esperado tras su ejecución. Esta comprobación se hará por medio de una tabla donde se indicará, por medio de un ✓, si esa prueba ha dado el resultado que se esperaba. No se han adjuntado imágenes para evitar recargar el documento, pero algunas de ellas pueden verse en *6.3 – MODO DE USO*.

PRUEBA	RESULTADO
PV-01	✓
PV-02	✓
PV-03	✓
PV-04	✓
PV-05	✓
PV-06	✓
PV-07	✓
PV-08	✓
PV-09	✓
PV-10	✓
PV-11	✓
PV-12	✓
PV-13	✓
PV-14	✓
PV-15	✓
PV-16	✓
PV-17	✓
PV-18	✓
PR-01	✓
PR-02	✓
PR-03	✓
PR-04	✓
PR-05	✓
PR-06	✓
PR-07	✓
PR-08	✓
PR-09	✓
PR-10	✓
PR-11	✓
PC-01	✓
PE-01	✓
PE-02	✓
PE-03	✓
PE-04	✓

Tabla 117: Análisis de los resultados de las pruebas

8 – Planificación y presupuesto

En este apartado se detalla la planificación estimada de realización del proyecto, se describe el cálculo de los costes y beneficios que han surgido del mismo y su impacto socioeconómico.

8.1 – Planificación

La planificación de este proyecto se ha llevado a cabo en función de dos factores. En primer lugar, la fecha de entrega del Trabajo Fin de Grado en su primera convocatoria del curso 2016-2017, comprendida entre el 20 y el 27 de febrero de 2017. Y, en segundo lugar, por el número de créditos de la asignatura Trabajo Fin de Grado, establecido en 12 créditos ECTS, donde cada ECTS equivale a 25 horas de trabajo del alumno, por lo cual la duración total de la asignatura es de 300 horas. Calculando unas 15 horas de trabajo semanal, la duración del proyecto debe ser de 20 semanas. De esta forma, dejando las 2 semanas anteriores a la entrega como tiempo de revisión, la fecha de inicio del proyecto tendrá lugar el 19 de septiembre de 2016, y la fecha prevista de finalización, el 3 de febrero de 2017.

La planificación del proyecto se ha dividido en las siguientes fases:

- **Documentación**.
- **Propuesta de proyecto**.
- **Estudio de la tecnología**.
- **Análisis**.
  - Valoración y elección de la solución.
  - Análisis de requisitos.
- **Diseño**.
  - Diseño de la arquitectura y la base de datos.
  - Diseño de casos de uso.
  - Diseño de clases.
  - Diseño de secuencia.
- **Implementación**.
- **Implantación**.
  - Obtención de datos.
  - Instalación del sistema.
  - Modo de uso.
- **Evaluación**.
  - Plan de pruebas.
  - Análisis de resultados.
- **Planificación**.

El siguiente diagrama de Gantt refleja cronológicamente la planificación de cada una de estas fases:

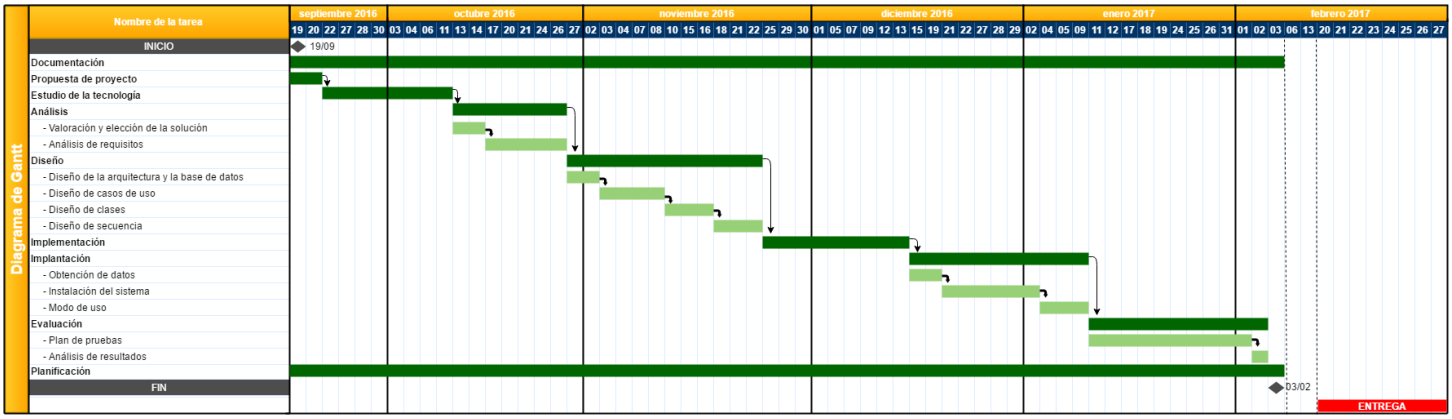

Ilustración 33: Diagrama de Gantt

8.2 – Presupuesto

Para calcular el coste de este proyecto debemos tener en cuenta varios factores. Lo primero es calcular el coste asociado al personal a partir de la planificación anterior y el sueldo medio de un trabajador. Después se calculan los costes asociados al material y los costes indirectos, y se suman a los costes del personal. Por último, calculamos el margen de riesgo y el de beneficio, y obtenemos el precio final del proyecto.

8.2.1 – Coste de personal

Corresponde al coste asociado al salario de los trabajadores que se encargan de la realización del proyecto. Para este proyecto, el perfil de trabajador que cuenta con las características necesarias para su realización es el Administrador de Bases de Datos. El salario de este perfil se ha obtenido a partir del salario medio de un profesional sin experiencia en el mercado laboral actual.

Las cotizaciones correspondientes a la cuota patronal se obtienen del salario base, es decir, el salario bruto del trabajador. No se incluyen pagas extras, pues se encuentran prorrateadas en el salario mensual.

Estos datos han sido obtenidos gracias al Departamento de Recursos Humanos de Everis Spain, S.L.U y a la página web de la Seguridad Social [40].

SALARIOS	
Salario bruto anual	26.556,00 €
Salario bruto mensual	2.213,00 €

CUOTA PATRONAL		
Contingencias comunes	23,6%	522,27 €
Desempleo	5,5%	121,72 €
Formación profesional	0,6%	13,28 €
FOGASA	0,2%	4,43 €
Cuota patronal mensual		**661,70 €**

PLANIFICACIÓN DEL PROYECTO		
Duración del proyecto	(horas)	300
Duración del proyecto	(semanas)	20
Horas de trabajo	(semana)	15
Horas de trabajo	**(mes)**	**60**

COSTE TOTAL DEL PERSONAL	
Coste mensual del trabajador	2.874,70 €
Coste de la hora trabajada	47,92 €
Coste total	**14.373,50 €**

Tabla 118: Coste de personal

8.2.2 – Coste del material

Corresponde al coste de los equipos donde se va a implantar el sistema, cuyas especificaciones están detalladas en *3.7 – ENTORNO OPERACIONAL*, y a las herramientas utilizadas para el desarrollo del proyecto. El precio de estos equipos se ha obtenido consultando la página web de uno de los proveedores de estos productos. Los precios vienen con IVA incluido.

COSTE DEL MATERIAL	
Servidor	494,00 €
Licencia Cassandra	0,00 €
Coste total	**494,00 €**

Tabla 119: Coste del material

8.2.3 – Costes indirectos

Corresponde a los costes asociados al material utilizado durante el desarrollo del proyecto y a otros posibles costes asociados al proyecto, como los costes del lugar de realización del proyecto. Los precios vienen con IVA incluido.

Para el coste de los equipos físicos se asume una vida útil de 4 años, y para el software se asume una vida útil de 3 años. Tomando un año de 52 semanas, las 20 semanas que dura la realización del proyecto equivalen a un 9,6% de la vida útil del equipo y a un 12,82% de la vida útil software. De este modo, en los costes indirectos sólo aplica el tiempo que se ha usado el material, es decir, el tiempo de amortización.

Los costes asociados al proyecto son la factura de alquiler del local donde se realizará el proyecto, que incluye agua y luz.

EQUIPOS FÍSICOS Y SOFTWARE DE DOCUMENTACIÓN	
Portátil Lenovo Thinkpad E550	668,99 €
TV Philips 22PFH4000 (como segunda pantalla)	149,00 €
Impresora HP Deskjet 3732	70,00 €
Microsoft Office Home & Business 2016	279,00 €

AMORTIZACIÓN		
Portátil Lenovo	9,6%	64,22 €
TV Philips	9,6%	14,30 €
Impresora HP	9,6%	6,72 €
Microsoft Office	12,82%	35,77 €
Total		**121,01 €**

OTROS COSTES	
Alquiler del local	500,00 €

COSTES INDIRECTOS TOTALES	
Coste total	**621,01 €**

Tabla 120: Costes indirectos

8.2.4 – Precio final

El precio final del proyecto es la suma de todos los costes, a los que se les aplica un margen de riesgo del 10% y un margen de beneficio del 15%.

COSTES TOTALES	
Coste de personal	14.373,50 €
Coste del material	494,00 €
Costes indirectos	621,01 €
Subtotal	**15.488,51 €**

Margen de riesgo	10%	1.548,85 €
Margen de beneficio	15%	2.323,28 €
Precio final		**19.360,64 €**

Tabla 121: Precio final del proyecto

8.3 – Impacto socioeconómico

El impacto socioeconómico se refiere al impacto que genera la realización del proyecto en el entorno. Dicho impacto puede tener repercusiones positivas o negativas, por lo que vamos a estudiarlo para determinar si genera un impacto positivo o un impacto negativo. La realización de este proyecto tiene un impacto tanto en el entorno académico, con la documentación de elección de BBDD NoSQL, como en el entorno de la Universidad, con el sistema elegido.

Primero vamos a revisar el impacto que puede tener la documentación sobre el entorno académico. Dado que es una documentación orientada a ayudar al usuario sobre bases de datos NoSQL, podría ser utilizado en cualquier aplicación que contara con un sistema NoSQL, para determinar cuál de ellos usar o para reafirmar por qué se ha escogido un determinado sistema. Puesto que la tecnología NoSQL es algo que aún está en crecimiento, creo que la existencia de esta documentación ayudaría a que la gente se interesara más por usar esta tecnología, puesto que no sería necesario informarse antes completamente de todos los posibles sistemas que existen, dado que ya tendrían una base de referencia sobre la que guiarse e informarse sólo de aquellos tipos que les interesen más.

A continuación, vamos a revisar las características del sistema para definir sus beneficios y sus inconvenientes:

- **Dar solución a la necesidad de más usuarios**. Dado que el sistema se va a utilizar en su mayor parte para consultas de los usuarios sobre la información de los miembros del departamento, la alta velocidad de respuesta de los sistemas NoSQL ante múltiples peticiones al mismo tiempo permite que se dé respuesta a más usuarios a la vez.
- **Estar siempre disponible para los usuarios**. Una de las características de Cassandra es que prima la disponibilidad sobre la consistencia. Esto podría ser un inconveniente si realizáramos muchas modificaciones diarias, puesto que cada usuario vería un dato distinto al hacer una consulta. Sin embargo, aquí es una ventaja, pues al no haber muchas modificaciones los datos siempre son los mismos, y no necesitamos mantener su consistencia; sin embargo, sí que necesitamos que el sistema esté siempre disponible por si algún usuario desea consultar algún dato en cualquier momento.
- **Adaptarse fácilmente al entorno**. Otra de las características de los sistemas NoSQL como Cassandra es la escalabilidad horizontal. Gracias a esta propiedad, el sistema puede adaptarse al crecimiento de la Universidad sin perder su eficacia.

Además de esto, considero que el proyecto en sí como conjunto tendría también un impacto sobre la sociedad. Por ejemplo, PYMEs recién creadas que se decantaran por las nuevas tecnologías NoSQL, o las que quisieran cambiar sus actuales sistemas de almacenamiento basado en BDR por uno no relacional, podrían usar este proyecto para hacer que sus trabajadores aprendieran más rápido sobre esta tecnología, o para que les generaran un informe sobre qué sistema va mejor para los datos que poseen. De esta forma, se ahorrarían el tiempo de tener que investigar por su cuenta qué tipos hay y qué características tiene cada uno, lo que se traduciría en un menor coste para la empresa.

Por último, dado que estos impactos son positivos, considero el impacto socioeconómico general del proyecto también positivo.

9 – Conclusiones y trabajos futuros

Este último apartado se divide en dos partes. En la primera se hace una pequeña reflexión a modo de conclusión sobre el producto desarrollado, el proceso de realización del proyecto y una reflexión personal sobre lo que ha sido el Trabajo Fin de Grado; y en la segunda, los trabajos futuros que podría generar este proyecto.

9.1 – Conclusiones

Las conclusiones de este proyecto abarcan todo lo que ha dado de sí el mismo, desde la experiencia personal y profesional que me ha aportado su realización hasta su posible utilidad en el mundo académico.

9.1.1 – Producto

Una vez concluido, podemos decir que se han cumplido satisfactoriamente los objetivos que definimos al principio del proyecto en el apartado *1.2 – OBJETIVOS*, tanto por parte del documento como por parte del sistema elegido, Cassandra.

Por parte del documento se cumplen los siguientes objetivos:

- ✓ Estudiar las posibles alternativas que pueden valer como sustitución del sistema LDAP, y compararlas con la solución propuesta (*2.2 – ESTUDIO DE POSIBLES SOLUCIONES*).
- ✓ Estudiar y analizar la diversidad de sistemas de gestión NoSQL que existen en el mercado actualmente (*2.1 – INTRODUCCIÓN*).
- ✓ Ofrecer al usuario una guía de elección de sistemas NoSQL (*2.3 – COMPARATIVA)*.
- ✓ Elegir, utilizando la guía de elección, uno de los sistemas estudiados como sustituto del LDAP (*3.4 – ELECCIÓN DE LA SOLUCIÓN*).

Y por parte de Cassandra se cumplen los siguientes objetivos:

- ✓ El sistema debe ser capaz de realizar las mismas tareas que el LDAP, como son añadir, modificar, borrar y buscar usuarios.
- ✓ El sistema debe ser capaz de realizar otras tareas, como la de insertar usuarios con distintos atributos, ya sea de uno en uno o desde un archivo.
- ✓ El sistema debe poder implantarse en los equipos de los técnicos del Laboratorio.
- ✓ El sistema debe permitir sacar los datos a un archivo siempre que se quiera.
- ✓ El sistema debe permitir la gestión de los usuarios que pueden acceder a él.
- ✓ El sistema debe permitir la gestión del acceso de forma remota.
- ✓ El sistema deberá ser de licencia gratuita y código abierto.
- ✓ El mantenimiento del sistema deberá ser poco costoso en horas por persona y en coste de equipos.

9.1.2 – Proceso

Sobre las conclusiones asociadas al proceso me gustaría destacar un par de problemas que, aunque a priori parecía que iban a dificultar el cumplimiento de los plazos establecidos para la realización del proyecto, acabaron resolviéndose con celeridad.

Siguiendo el mismo orden que el documento, empezaré por comentar que, debido a mi nulo conocimiento sobre los sistemas NoSQL, la fase de Estudio de la tecnología, y por tanto el proyecto, se ha alargado más de lo debido. La duración de este proyecto se podría haber acortado considerablemente si se hubiera conocido la tecnología de antemano.

La primera dificultad que me surgió fue al obtener los datos almacenados en el LDAP. Como ya comenté en el apartado *6.1 – EXTRACCIÓN DE LOS DATOS*, obtener los datos a través del código HTML no fue mi primera opción, pues era largo y tedioso, al menos más largo y tedioso que obtener un archivo por parte de la Universidad. Por lo tanto, aconsejado por mi tutor, decidí obtener los datos de otra manera. Sin embargo, el formato del código HTML de las páginas web del Departamento de Informática no facilitaba la extracción de los datos, pues tenía que buscar ciertas cadenas de caracteres dentro del código, y a menudo había dos cadenas iguales y se recogían datos que no eran correctos. A pesar de eso, me las ingenié para acotar la zona de búsqueda y recoger sólo la información que necesitaba. Pero cuando pensé que ya tenía todos los datos, vi que el correo no se almacenaba en el código de manera clara, sino con símbolos HTML, para que, en caso de que alguien quisiera buscar los correos (como estaba haciendo yo) para propósitos publicitarios como spam, no pudiera obtenerlos en claro. Después de hablarlo con mi tutor, se nos ocurrió la idea de usar funciones PHP para, a través de la cadena cifrada que contenía el correo, obtenerlo en claro.

El otro problema que me ha surgido no ha sido tanto un problema que tuviera que resolver, sino una falta de conocimiento sobre Cassandra. Cuando arrancaba la máquina virtual o cuando reiniciaba Cassandra después de modificar los ficheros de configuración, intentaba acceder a la misma con el comando *cqlsh*, pero me daba un error de conexión y no podía acceder. Esto me sucedía unas veces sí y otras no, por lo que no le encontraba explicación. Por suerte, más tarde me di cuenta, consultando por la web y mediante prueba y error, que Cassandra tardaba un tiempo en iniciarse/reiniciarse, por lo que, si intentaba acceder demasiado pronto, me daba error de conexión.

Por último, quería comentar que este proyecto podría haberse paralelizado en algunos puntos, si se hubiera contado con más gente trabajando en él. Suponiendo que se conoce de antemano la tecnología a usar, las fases de Implementación e Implantación podrían haberse hecho al mismo tiempo, y la fase de Evaluación se podría haber dividido entre el número de trabajadores para acortar su tiempo de realización. De esta forma se reduciría el camino crítico, es decir, la duración total del proyecto.

9.1.3 – Personales

A nivel personal, estoy muy orgulloso del trabajo que he realizado. Durante estos 5 meses de realización del proyecto he aprendido muchas cosas, y afianzado otras que ya tenía aprendidas. Además, creo que he cumplido el objetivo que me propuse al empezar con el proyecto, que es aprender más sobre lo que peor se me ha dado de la carrera, como es el tema de bases de datos.

Sin embargo, en el fondo sé que no habría podido hacer ni la mitad de lo que he hecho sin la formación que he recibido durante mi carrera universitaria. Aunque de un modo general todas las asignaturas me han aportado algo, he aquí las que más me han ayudado a realizar el Trabajo Fin de Grado:

- **Ficheros y Bases de Datos**. Al ser este un proyecto relacionado con las bases de datos, me parecía lo más lógico empezar por la asignatura de la carrera más afín al tema. Aunque en esta asignatura se ven BDR y yo he tratado el tema de las no relacionales, los conocimientos adquiridos en esta asignatura me han valido para entender mejor cómo funcionan los sistemas NoSQL, además de poder utilizar los conocimientos sobre lenguaje SQL que aprendí con ella para ejecutar operaciones en Cassandra.
- **Ingeniería del Software y Dirección de Proyectos de Desarrollo de Software**. Gracias a esta asignatura me ha sido más fácil realizar la documentación de la memoria, además de que he podido recurrir a ella para realizar el análisis de requisitos y los diagramas UML. Aun así, creo que estas asignaturas serían más útiles en el último año de carrera, pues estarían más cerca del Trabajo Fin de Grado, que es donde puedes poner en práctica los conocimientos aprendidos en ellas.
- **Interfaces de Usuario**. Gracias al conocimiento sobre HTML que aprendes en esta asignatura, me ha sido más fácil indagar en el código de las páginas web del Departamento de Informática para obtener los datos que necesitaba.
- **Sistemas Operativos, Arquitectura de Computadores, Diseño de Sistemas Operativos y Sistemas Distribuidos**. Gracias al uso del lenguaje C en estas asignaturas he podido realizar unos programas en dicho lenguaje, que me han permitido obtener los datos que necesitaba.

Por último, me gustaría comentar lo que muchas veces digo cuando, en una entrevista de trabajo, por ejemplo, me preguntan qué es lo que más valoro de mi paso por la universidad. Una vez que sales al mundo laboral, por mi experiencia personal, me he dado cuenta de que, independientemente de los conocimientos específicos que te enseñen en la misma, a lo que sí te enseñan es a enfrentarte a un problema y a resolverlo por tus propios medios, dándote cabezazos con él hasta que lo solucionas. Gracias a ello he podido aprender una tecnología totalmente nueva para mí, como es el NoSQL, sin necesidad de un maestro a mi lado que me fuera enseñando. Además, la diversidad de lenguajes de programación que te enseñan me ha permitido aprender más rápidamente uno nuevo, como ha sido PHP.

9.2 – Trabajos futuros

En este apartado se encuentran los posibles trabajos que podrían surgir como consecuencia de la realización de este proyecto, y que, la mayoría, no han podido ser implementados por falta de tiempo.

Datos de otros Departamentos

En este proyecto sólo se han usado los datos del Departamento de Informática para insertarlos en la base de datos elegida. Un posible trabajo futuro sería obtener de algún modo los datos de todos los departamentos de la Universidad Carlos III e insertarlos en la base de datos. Para ello, habría que crear un *keyspace* por cada departamento, para seguir el esquema de la base de datos indicado en *4.2 – DISEÑO DE LA BASE DE DATOS*. De este modo, se tendría toda la información del personal de la universidad accesible en el mismo sistema.

Alojar datos en la nube

En el análisis de requisitos se definió que una vez al mes se debían exportar los datos a un archivo, para salvaguardarlos en caso de que fallara el sistema. Sin embargo, si almacenamos dicho archivo en un solo equipo, y ese equipo sufre un accidente, ya sea una rotura o accidente en el lugar en el que se encuentra el equipo, como un incendio o inundación de la sala, los datos se perderían. Por tanto, sería recomendable guardar ese archivo en la nube, de modo que los datos se mantuvieran a salvo en caso de que el equipo sufra algún daño.

Seguridad

Una vez implementado el sistema, se deberían realizar ciertos pasos para configurar la seguridad del mismo.

Uno de ellos sería configurar el acceso remoto de Cassandra para permitir el acceso sólo con usuario y contraseña. De esta forma impediríamos el acceso no autorizado al sistema. También podríamos configurar el archivo de *iptables* para que sólo se acceda desde direcciones de la red de la universidad o direcciones específicas de fuera de la red de la universidad.

Otro paso importante para mejorar la seguridad del sistema sería modificar el archivo *iptables* para que se acepten todos los paquetes de salida, pero sólo ciertos paquetes de entrada, o cifrar todas las conexiones entre el equipo local y el remoto.

Página web

En el análisis de requisitos se definió que debía haber una página web donde estuviera la ayuda para la elección de una base de datos NoSQL. Para crear dicha página web, he creado un archivo en código HTML que contiene dos imágenes, cada una con un árbol de decisión, y debajo de cada árbol, la explicación de las características o tipos que se pueden elegir y las especificaciones de cada base de datos NoSQL elegible, según el camino que se escoja. Este archivo puede encontrarse en la carpeta *src* del repositorio [33] de GitHub.

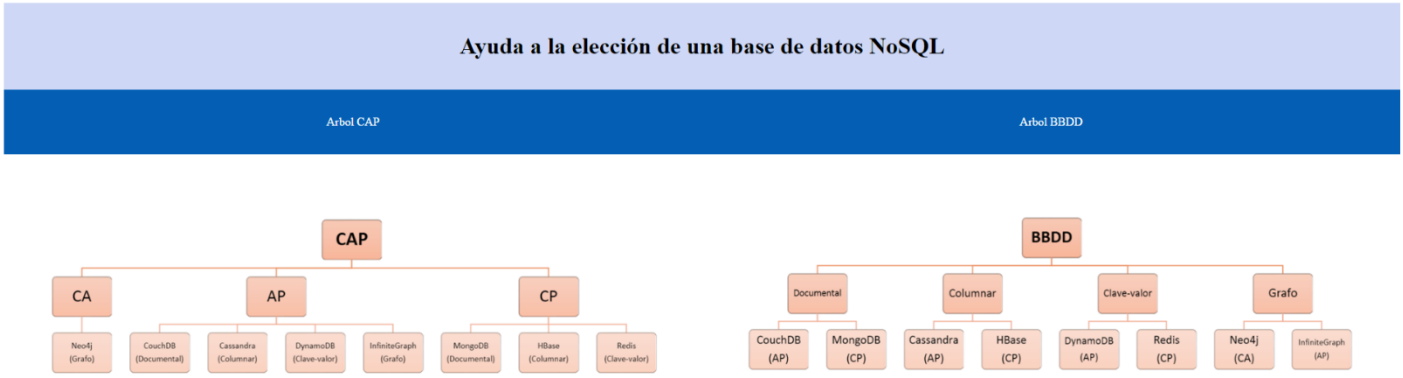

Ilustración 34: Página web con la ayuda de elección de base de datos

La función definida como *show_doc* también habría que crearla, pues no es una función propia de Cassandra. Bastaría con que esta función accediera a un *keyspace* distinto al del creado para el Departamento de Informática, donde se alojaría la ruta donde está el archivo o el código del archivo en sí, y devolvería esta información. Esta funcionalidad no se ha implementado por falta de tiempo.

Importar los datos del Dpto. de Informática a la agenda del móvil

Este posible trabajo surgió a raíz de la necesidad de mi tutor, Alejandro Calderón, de tener a los contactos del Departamento de Informática fácilmente accesibles desde el móvil. Este trabajo sí se ha podido desarrollar debido a que no suponía una carga de trabajo muy alta, ya que se contaba con las bases necesarias para su realización, y sólo hacía falta hacer unas pequeñas modificaciones.

Para su desarrollo, se ha partido de los programas desarrollados para la obtención de los datos del Dpto. de Informática. Después de estudiar cómo se podían importar contactos desde un archivo a la agenda de Google, he descubierto que sólo necesitaba modificar cómo guardaba la información obtenida del departamento para adaptarla al formato de la agenda de Google. Para ello, he exportado los contactos de mi agenda en un archivo con formato *csv* y he copiado la cabecera a otro archivo *csv*. En ese archivo he guardado después el nombre, apellidos, email y teléfono de cada miembro del departamento. Por último, he importado ese archivo a la agenda de Google para comprobar que estaba en el formato correcto.

Anexo

Apéndice I – Acrónimos y definiciones

Los siguientes términos han sido sacados de estas referencias: [41] y [42].

- **ANSI-SPARC**. Arquitectura de 3 niveles para las bases de datos aprobada por el Instituto Nacional Estadounidense de Estándares (ANSI) y el Comité de Requisitos y Planificación de Estándares (SPARC).
- **API** (*Application Programming Interface*). Conjunto de convenciones internacionales que definen cómo debe invocarse una determinada función de un programa desde una aplicación. Cuando se intenta estandarizar una plataforma, se estipulan unos APIs comunes a los que deben ajustarse todos los desarrolladores de aplicaciones. Herramientas de programación para rutinas, protocolos y software.
- **BBDD** (Base de Datos). Conjunto de datos relacionados que se almacenan de forma que se pueda acceder a ellos de manera sencilla, con la posibilidad de relacionarlos, ordenarlos en base a diferentes criterios, etc.
- **BDR**. Base de Datos Relacional.
- **Bit**. Es la unidad de información más pequeña. Puede tener sólo dos valores o estados: 0 o 1.
- **Byte**. Ocho bits que representan un carácter. Unidad básica de información con la que operan los ordenadores.
- **CLI** (*Command Line Interface*). Tipo de interfaz para manipular un programa o sistema operativo con instrucciones escritas. También llamado Consola en Windows o Terminal en Linux.
- **Clic**. También conocido como Click, es la acción de presionar algún botón del ratón o mouse.
- **Descomprimir**. Proceso inverso a comprimir. En general, la información comprimida debe primero descomprimirse para que pueda ser accedida, leída o modificada.
- **DDoS**. Ataque de denegación de servicio. Consiste en saturar de peticiones el servidor para hacerlo no funcional.
- **Diagrama de caso de uso**. Sirven para especificar la funcionalidad y el comportamiento de un sistema mediante su interacción con los usuarios y/u otros sistemas. O lo que es igual, un diagrama que muestra la relación entre los actores y los casos de uso en un sistema. Se utilizan para ilustrar los requerimientos del sistema al mostrar cómo reacciona una respuesta a eventos que se producen en el mismo.
- **Diagrama de flujo**. Diagramas que utilizan símbolos para representar y especificar detalles algorítmicos de un proceso. En otras palabras, representan gráficamente los pasos de un proceso.
- **Dirección IP**. La forma estándar de identificar un equipo que está conectado a Internet, de forma similar a como un número identifica un teléfono en una red telefónica. La dirección IP consta de cuatro números separados por puntos y cada uno es menor de 256; por ejemplo 192.200.44.69.

- **Disco duro**. Un disco duro es un tipo de disco magnético para el almacenamiento duradero de datos.
- **ECTS** (*European Credit Transfer System*). Sistema Europeo de Transferencia de Créditos. Es el sistema adoptado por todas las universidades del Espacio Europeo de Educación Superior (EEES) para garantizar la homogeneidad y la calidad de los estudios que ofrecen.
- **Escalabilidad**. Propiedad deseable en un sistema, red o proceso que indica su habilidad para poder hacerse más grande sin perder calidad en sus servicios.
- **GB** (*Gigabyte*). Unidad de almacenamiento que representa 1.000.000.000 (mil millones) de bytes.
- **Gmail**. Servicio de Google que permite crear cuentas de correo electrónico para usar por web.
- **Hoja de cálculo**. Tipo de aplicación que es usada en análisis y cálculos matemáticos, que permite trabajar sobre una matriz compuesta por celdas o casillas.
- **HTML** (*HyperText Markup Language*). Es el lenguaje estándar para describir el contenido y la apariencia de las páginas en el WWW.
- **Instantánea**. En informática, una instantánea es la situación de un sistema en un momento determinado.
- **JOIN**. Sentencia SQL que permite combinar registros de dos o más tablas en un modelo relacional.
- **JSON** (*JavaScript Object Notation*). Es un formato ligero para el intercambio de datos. Su formato es: {“clave1”: “valor”, “clave2”: “valor”, …}.
- **JSP** (*Java Server Page*). Se refiere a un tipo especial de páginas HTML, en las cuales se insertan pequeños programas que corren sobre Internet (comúnmente denominados scripts), que se procesan en línea para finalmente desplegar un resultado final al usuario en forma de HTML.
- **LDAP** (*Lightweight Directory Access Protocol*). Protocolo ligero de acceso a directorios.
- **Linux**. Versión bajo la licencia GPL/GNU (que permite la copia y distribución junto al código fuente y sólo se paga el "medio físico") del conocido sistema operativo UNIX. Es un sistema multitarea multiusuario para PC´s.
- **Login**. Proceso con el que se denomina el comienzo de una sesión en un sistema informático, usualmente compuesto por el pedido de un nombre de usuario (username) y una clave (password), como medio fehaciente de autentificar la identidad del usuario.
- **Mapear**. Técnica para convertir datos de un formato a otro distinto.
- **Máquina Virtual**. Software que simula el funcionamiento de una computadora real para poder ejecutar programas.
- **MB** (*Megabyte*). Unidad de almacenamiento que representa 1.000.000 (un millón) de bytes.
- **MIT** (*Massachussets Institute of Technology*). Universidad privada fundada en 1860 en Cambridge, Massachusetts (Estados Unidos).
- **Modelo Entidad-Relación**. Es un tipo de modelo de datos conceptual de alto nivel que se emplea en el diseño de las bases de datos relacionales.
- **Modelo Relacional**. Representa la segunda generación de los SGBD. En él, todos los datos están estructurados a nivel lógico como tablas formadas por

filas y columnas, aunque a nivel físico pueden tener una estructura completamente distinta.

- **NoSQL** (*Not Only Structured Query Language*). Conjunto de SGBD que pueden o no usar el lenguaje SQL para acceder a la base de datos.
- **P2P** (*Peer-to-peer*). Red descentralizada que no tiene clientes ni servidores fijos, sino que tiene una serie de nodos que se comportan simultáneamente como clientes y servidores de los demás nodos de la red. Cada nodo puede iniciar, detener o completar una transacción compatible.
- **PHP** (*Preprocessed Hypertext Pages*). Es un lenguaje de scripting embebido en HTML. Mucha de su sintaxis es tomada de C, Java y Perl con un par de características adicionales únicas y específicas de PHP. El propósito del lenguaje es permitir que los desarrolladores web escriban páginas generadas dinámicamente con rapidez.
- **PYME**. Pequeña Y Mediana Empresa.
- **RAM** (*Random Access Memory*). Tipo de memoria donde la computadora guarda información para que pueda ser procesada más rápidamente. En la memoria RAM se almacena toda información que está siendo usada en el momento.
- **Script**. Pequeño programa escrito para un intérprete de comandos con algún lenguaje de scripts como puede ser bash, sh, python, perl, etc.
- **SGBD** (Sistema Gestor de Bases de Datos). Es una aplicación que permite a los usuarios definir, crear y mantener la base de datos, y proporciona acceso controlado a la misma.
- **Sistema**. En informática, este término utilizado sin otra palabra que lo adjetive designa un conjunto de hardware y software específico. Conjunto de elementos interrelacionados y regidos por normas propias, de modo tal que pueden ser vistos y analizados como una totalidad. El sistema se organiza para producir determinados efectos, o para cumplir una o varias funciones.
- **Solaris**. Variante del sistema operativo Unix desarrollada por Sun Microsystems.
- **SO/SSOO** (Sistema Operativo/Sistemas Operativos). Programa fundamental que administra los demás programas en un ordenador.
- **Spam** (correo basura). Mensajes no solicitados, habitualmente de tipo publicitario, enviados en grandes cantidades (incluso masivas) que perjudican de alguna o varias maneras al receptor. Aunque se puede hacer por distintas vías, la más utilizada entre el público en general es la basada en el correo electrónico.
- **SQL** (*Structured Query Language*). Es un estándar en el lenguaje de acceso a bases de datos.
- **Streaming**. Consiste en una tecnología utilizada para permitir la visualización y la audición de un archivo mientras se está descargando, a través de la construcción de un buffer por parte del cliente, una vez que este se ha conectado al servidor, el buffer del cliente se va llenando de la información descargada y se va reproduciendo en el ordenador
- **TB** (*Terabyte*). Unidad de almacenamiento que representa 1.000.000.000.000 (un billón) de bytes.

- **Ubuntu**. Es una distribución Linux que ofrece un sistema operativo predominantemente enfocado a computadoras de escritorio, aunque también proporciona soporte para servidores. Es una de las más importantes distribuciones de GNU/Linux a nivel mundial.
- **UML** (*Unified Modeling Language*). El lenguaje para modelamiento unificado (UML), es un lenguaje para la especificación, visualización, construcción y documentación de los artefactos de un proceso de sistema intensivo.
- **UNIX**. Es una familia de sistemas operativos tanto para ordenadores personales como para mainframes. Soporta gran número de usuarios y posibilita la ejecución de distintas tareas de forma simultánea (multiusuario y multitarea). Su facilidad de adaptación a distintas plataformas y la portabilidad de las aplicaciones (está escrito en lenguaje C) que ofrece hacen que se extienda rápidamente.
- **URL** (*Uniform Resource Locator*). Localizador Universal de Recursos. Sistema unificado de identificación de recursos en la red. Las direcciones se componen de protocolo, FQDN y dirección local del documento dentro del servidor. Permite identificar objetos WWW, Gopher, FTP, News, etc.
- **Windows**. Es el nombre del popular entorno (no es un sistema operativo y no es una aplicación) software creado por Microsoft. Su novedad es el uso de diferentes pantallas que se superponen, denominadas ventanas, para mostrar distintos tipos de información.
- **WWW** (*World Wide Web*). Telaraña o malla mundial. Sistema de información con mecanismos de hipertexto creado por investigadores del CERN. Los usuarios pueden crear, editar y visualizar documentos de hipertexto.

Apéndice II – Resumen del proyecto en inglés

Introduction

Nowadays, NoSQL technology is constantly expanding owing to its improvements in comparison with SQL traditional systems. One of these improvements is the ability these systems have to manage a huge amount of data and how quickly is access to this information. This is the reason why the number of companies using these databases is increasing recently.

However, there are too many NoSQL systems management, and no recommendation guide about which one to use in each moment. That is the goal of this project, to be a manual about what system you have to use depending on your situation.

First of all, I am going to study NoSQL term and different Database Management Systems (DBMS). Then, I am going to use data taken from LDAP of the computer science department, and I will store them on a NoSQL database selected, for the purpose of replacing the current LDAP system. This is the other goal of this project.

Motivation and objectives of the product

The main reason why I decided to do this project was because my knowledge about databases is not very deep, considering that it has not been a strong point in my degree course. Therefore, when people from the Computer Science and Engineering Department notified me that they were going to manage without the LDAP system they were currently using, I proposed to them to move the information to a NoSQL database.

Nevertheless, as I have said before, the number of NoSQL systems is too high, so before migrating the data I needed to know the destination system. This way, I made the decision to add a study to my project in which I compare several NoSQL DBMS so as to decide which of them would be more appropriate to the subsequent installation.

The purpose of this project is to propose an alternative storage system to the current LDAP in order to store the Computer Science Department's data. Thus, it is needed to study in depth NoSQL technology so as to reduce the possible alternatives with the aim of choosing the most appropriate one. To do this, some objectives have been set:

- Studying the possible alternatives.
- Studying and analyzing the range of NoSQL systems.
- Offering the user a manual to choose a NoSQL system.
- Choosing the best NoSQL system to replace LDAP.

Memory Structure

At this point, I am going to explain how the document is structured. It is divided into the following sections:

- **Introduction.** This section contains a brief introduction about the subject I am dealing with and what are the next steps to follow.
- **Art state.** This section contains a description about the NoSQL term, a study of different solutions I could have chosen instead of a NoSQL database and a comparative between different NoSQL databases.
- **Analysis.** This section contains the lifecycle of the project, the solution chosen and its features, the requirements the solution must achieve, and the environment where the solution will run.
- **Design.** This section contains the system architecture, the database design, the use cases, sequence and class diagrams, and, lastly, the system interface.
- **Implementation.** This section contains the operation of the developed code.
- **Establishment.** This section contains all the information about the process of obtaining data which are going to be used later and the database installation and its ways to be used.
- **Evaluation.** This section contains the system tests.
- **Planning and budget.** This section contains the planning of the project and its budget.
- **Conclusion.** This section contains the final conclusions of the Project and the possible future projects.
- **Attached.** This section contains the appendices of the project.
- **Bibliography.** This section contains all the bibliographic sources I have used in this project.

Art state

The term NoSQL was created in 1998 and refers to a relational open-source database that did not use SQL as query language, but it was not until 2009 when this term was associated with the new non-relational databases were appearing by then.

Until then, SQL was the most common programming language used by database systems of companies, which could be augmented if they need more calculation capacity. However, with the arrival of the web and cloud services, SQL databases are not capable to provide the service to a huge amount of users consulting the same information at the same time (Google, Facebook, Amazon).

Thus, even though NoSQL lose the possibility of doing JOINs or operations between different data collections, NoSQL systems try to solve this problem by proposing a more adaptable data storage. This, coupled with its scalability, especially horizontally with its ability to put more nodes doing the same task to decrease time, make it more competent for queries between a big data volume.

NoSQL term is not about databases that do not use SQL as query language, is about databases that do not use SQL as the only query language. These databases can use other types of query languages, like JSON or GQL. The basic difference between SQL and NoSQL databases is that the latter does not use a relational model to store data, and the other does. So, as Carlo Strozzi, who invented this term, said, these databases should be called NoRel.

Now that we know the meaning of NoSQL, its origin and its features, you must know that there are many NoSQL systems, divided in multiple groups according to their functionality. Some of them, like XML, Object Oriented, Grid & Cloud, Multimodel or Multidimensional, among others, will not be studied in this project. Those we are going to study are divided into the following groups:

- **Document databases.** They store information in a semi-structured way, in documents whose format is JSON or BSON. These documents are similar to relational database records, but more adaptable; that is, two documents may have different attributes and to be both in the same database. Documents are addressed by a unique key with which information is recovered, but these databases also have an API that allows the recovery at information associated with a specific field or fields. Due to their performance, flexibility and ease of use, document databases are probably the most popular of the NoSQL databases.
  Examples: CouchDB and MongoDB.
- **Column Family databases.** They store information in columns, with a primary key to recover and update data. Each key can have more than one value. They are similar to relational databases, but using columns instead of records. These databases are designed for large volumes of data, high availability and read and write performance. Column Family databases run on clusters of multiple servers, so if your data is small enough to run with a single server, you should probably use a document or key-value database.
  Examples: Cassandra and HBase.

- **Key-value databases.** They store information by records that have a key and a value. The same as column family, key-value databases also have a primary key to recover and update data. However, as opposed to the first one, these databases have only one value for each key. If you want to recover information, you just have to search by key to obtain its value. These databases are well-suited to applications that have issues of massive writes of streaming.
  Examples: DynamoDB and Redis.
- **Graph databases.** They store information as a graph, employing nodes (represented as circles) and relationships (represented as arrows) to show the stored data. Nodes represent entities, which can hold any number of attributes, each one as a key-value pair, and can be tagged with labels corresponding to their different roles in your domain. Relationships represent the connection between two nodes. They always have a direction, a type, a start node and an end node, and, like nodes, relationships can have any properties. In spite of relationships are directed, they can always be navigated regardless of direction. These databases are especially useful in models with many relationships, because they are more efficient to queries where there are proximity relationships between data than to global queries.
  Examples: Neo4j and InfiniteGraph.

The problem here is, not even reducing the number of NoSQL databases to four, but being sure about which one is best for whatever we might need. This is why we need a guide to follow, in order to identify what system will be more effective for the data we have and the operations we need to do.

Other solutions

Despite the fact that a database is a good choice to store current LDAP data, it is not the only one I could have chosen. There are two other options that I have considered as possible solutions but, for different reasons, I have decided to reject in favour of a NoSQL database. These are the two options I have decided not to implement:

- **Relational Database**. A relational database follows the relational model, which is the most widespread model to implement databases. Moreover, it uses Structured Query Language to recover the stored information. Its advantages are that it guarantees consistency and integrity, and it offers a simple interface to manipulate data. Its disadvantages are that a lot of queries at the same time might slow down the system, and if you need to add a column to the model, you have to redo it.
- **Google contact**. It is a free tool to manage Google contacts that comes with any Gmail account. It is an online agenda that you can synchronise with all your devices and access everywhere. Its advantages are that you can gather your contacts by Department or Category and it has a search bar to look up easily users. Its disadvantages are that you can only store 25.000 contacts, and you

have to export contacts to a file in order for users to be able to add them to their Gmail accounts.

- **Excel in Google Drive**. Microsoft Excel is a spreadsheet developed by Microsoft. It is an application dedicated to logical and mathematical formulas, even though it can be used to keep any type of information in its cells. Its advantages are that you only need to access the URL to get the contacts file, and the data are human readable. Its disadvantages are that it has many restrictions, and if you want to insert or delete a user, you have to look up him previously.

Comparative study

To make this comparative study I am going to start from CAP Theorem. This theorem, also called Brewer Theorem in honor of Eric Brewer, its founder, says that any distributed system cannot guarantee at the same time as the next characteristics:

- **Consistency**. It is when every node can see the same value for a datum.
- **Availability**. It is when the system is always available to the user, whenever the request is received.
- **Partition tolerance**. It is when the system keeps processing requests in spite of some nodes are separated from the network.

Therefore, CAP Theorem only guarantees that two of these qualities will be fulfilled:

- **CA**. It guarantees consistency and availability, but not partition tolerance.
- **AP**. It guarantees availability and partition tolerance, but not consistency.
- **CP**. It guarantees consistency and partition tolerance, but not availability.

At this point, the user who reads this guide must think about what qualities he prefers for his system, and to choose two of them. Depending on what characteristics he chooses, there will be different options:

- **Consistency and Availability**. If the user chooses these two features, he can only select between a relational database or the only NoSQL option that guarantees these two features: Neo4j.
- **Availability and Partition tolerance**. If the user chooses these two features, he can select between CouchDB, if his information is stored in files with a similar structure to JSON format, Cassandra, if his information is stored in a column way, DynamoDB, if his information is stored in a key-value way, and InfiniteGraph, if his information is stored in a similar way to relational model.
- **Consistency and Partition tolerance**. If the user chooses these two features, he can select between MongoDB, if his information is stored in files with a similar structure to JSON format, HBase, if his information is stored in a column way, and Redis, if his information is stored in a key-value way.

Analysis

In this section, I am going to explain the next parts:

- **Lifecycle**. This part is about the lifecycle of the project. It is a project about a database, so that the stages of this lifecycle are: Analysis, Design, Establishment, Tests, Maintenance. It is a circular lifecycle, so after Maintenance will come Analysis again.
- **Regulatory framework**. This part is about the law associated with this project. There are three laws to apply to this project. The first one is the Protection of Personal Data Law, which says that it must be guaranteed privacy of the personal data of the UC3M staff. This project can guarantee this because that information is encoded with a symmetric cipher named AES before being stored in the database. The second law is Telecom General Law, which says that the client-server communication must be encoded. And the last law is the Intellectual Property Law, which says that the intellectual property of this project belongs to me because it is my name which is on the cover of this document. In addition, I have used free software tools like VirtualBox and Cassandra, whose licenses are free because these tools are distributed by GNU General Public License and Apache License.
- **Solution chosen**. This part is about the features the solution must guarantee and what possible solution does it. The solution must allow to insert, update, read and delete rows, to create, alter and drop tables, keyspaces and roles, to grant and revoke permissions and, finally, to be able to be installed in the Computer Science lab's computers. Among all the options I could have chosen, the best solution to guarantee these features is Cassandra.
- **Requirements**. This part is about the requirements the solution chosen must guarantee to satisfy client's necessities. These requirements are: to insert, update, read and delete rows, one by one and many at a time; to search the database by different attributes; to store data in an external file; to create, alter and drop roles; to grant and revoke permissions; to list roles and permissions; the starting of the solution will not have more than three commands.
- **Operational environment**. This part is about the computer where the solution was tested and the computer where the solution will be established. The first one is my own computer, a Lenovo ThinkPad with Windows 10, and the second one is the lab's computer, with Windows 7. Also, this part shows a schema about how the server would be accessed from outside the university network.

Design

In this section, I am going to explain the next parts:

- **System architecture**. Like all databases, the system has an ANSI-SPARC architecture. This is a three-level architecture: External Level, a user's view of the database, Conceptual Level, a way of describing what data is stored and how it is inter-related, and Internal Level, how the data is physically represented on the computer. Also, Cassandra has its own features: it is a distributed system, because it stores data in several nodes in a replicative way, it has lineal and horizontal scalability, because it is able to increase computer capacity if we add more nodes, and it establishes a P2P architecture.
- **Database design**. The database design has the next elements, starting for the higher: Cluster, Keyspace, Column-family, Row, Column, Key-value.
- **Use cases, class and sequence diagrams**. The use cases are: insert user, update user, delete user, insert file, update all, delete all, export data, read data, create role, update role, delete role, grant permissions, revoke permissions, list roles, list permissions, create keyspace, and create table. The classes are: Cassandra, which has the information about the created keyspaces and tables, and the system configuration files; and whose methods are the related ones to the system (create keyspace, create table, insert file and export data); User, which represents the data stored in the system, and whose methods are insert user, update user, delete user, drop all and read data; Role, which exists to manage the access to the system, and whose methods are create role, alter role, drop role and list roles; and Permission, which exists to manage the access to the system data, and whose methods are grant permission, revoke permission and list permissions. The sequence diagram is how the classes defined before interact.
- **System interface**. This part describes the user interface of Cassandra. This system was installed in a Linux distribution, so the user interface I am going to use is the Ubuntu terminal. However, like many others databases, there is a driver which allows you to connect Cassandra with a Java programming environment like Eclipse. I have tried this and it worked, but I decided to keep using the Ubuntu terminal.

Implementation

In this section, I am going to explain how the developed code used to get the data stored in the LDAP works. It is divided in different files. I will explain each one of them. These files are:

- **codigoHTML.txt**: This file is obtained after running the curl command in Cygwin64 for the UC3M Computer Science Department's staff website. In this file are contained the URLs of each member of the staff, in order to access them later to obtain the information of these members.
- **codigo1.c**: This file has been developed to get the URLs of each member of the staff from *codigoHTML.txt*, and store them in the file *salida.sh*.
- **salida.sh**: This file contains a string of characters with curl command, the URL you want to get the information from and the output file. After running this code, we will have a folder called *salidas* with a file for each member of the staff. These files contain all the information we want to get about the members.
- **codigo2.c**: This file has been developed to decode the staff emails, because in the beginning, these emails were coded with HTML codes in order to avoid spam to their owners. Therefore, this code stores in *codigoPHP.php* a string of characters for each member, which contains html_entity_decode() and the user email between the parenthesis. Later, these emails will be decoded by the output file.
- **codigoPHP.php**: This file contains a string of characters with the html_entity_decode command and the HTML codes to decode.
- **correos.txt**: This file is obtained after running *codigoPHP.php*, and it contains the decoded emails of the Computer Science Department's staff.
- **codigo3.c**: This file has been developed to obtain the staff information. To do this, it is necessary to get the emails from *correos.txt* and the remaining information, like name, surname or phone number, from *datosLDAP.txt*, in which it is necessary to look up all the fields in a certain way.
- **datosLDAP.txt**: This file is obtained after running *codigo3.c*, and it contains all the necessary information about the Computer Science Department's staff.
- **codigo3_toJSON.c**: This file has been developed to obtain the staff information in a specific way in order to JSON Generator website is able to save it in JSON format.
- **datosLDAP_JSON.txt**: This file is obtained after running *codigo3_toJSON.c*, and it contains all the necessary information about the Computer Science Department's staff in a suitable format for JSON Generator website.
- **generated.json**: This file is obtained from the output of JSON generator website.
- **result.csv**: This file is obtained from the output of Konklone website, which changes *generated.json* file to CSV format.

Establishment

In this section, I am going to explain all the steps associated with the performance test detailed in the next section. These steps are separated in three stages:

- **Data extraction**. This stage is about getting the information about the Computer Science Department´s staff in order to later store the data in the database. To do this, I have developed some code files, explained in the previous section, so as to get the information I need from the UC3M Computer Science Department's staff website. In first place, I have used the tool Cygwin64 to get the file *codigoHTML.txt*, which has the code of the UC3M website. Secondly, I have obtained all the Computer Science Department's staff websites by running *codigo1.c*, and I have obtained the information contained in these websites by running *salida.sh*. Finally, I have obtained all the information I needed by running *codigo3.c*.
- **Database installation**. This stage teaches how to install the system chosen. First of all, you have to install VirtualBox in your computer, and then create a new virtual machine with this program. When you have the virtual machine, you have to install Ubuntu on it. Finally, you have to follow the steps you will find on the internet to install Cassandra on your virtual machine.
- **Use mode**. This stage shows some commands you can use with Cassandra, like *create keyspace* or *create role*.

Evaluation

This section describes the test plan of the project. This plan guarantees that the requirements defined before are achieved. This plan is divided into four types of tests:

- **Validation tests**. Its goal is to confirm the system achieves the promised functionality. They test the requirements so as to be able to tell the client everything they asked is established.
- **Restriction tests**. Its goal is to verify the restrictions of the system are fulfilled. To do that, it is necessary to test the restriction in order to verify if that operation can or cannot be done, whatever its functionality may be.
- **Compatibility tests**. Its goal is to confirm if the system is able to be installed in different operative systems. To do that, I have installed the solution chosen in both operative systems, Windows and Linux.
- **Performance tests**. Its goal is to determine how fast a system is able to do a task in certain work conditions. To do that, I have used the Unix command *time*, which measures the time of an operation. I have only measured the next operations: insert, update, delete, read, because they will be the most used operations.

Planning and budget

This section is about the estimated planning of the project, the cost and profit calculation and its socio-economic impact.

- **Planning**. This planning has been developed according to two factors. The first one is the delivered time of the project, which is between February 20th and 27th. And the second one is the number of credits, 12, the subject TFG has. Each credit is equivalent to 25 student work hours, so that the subject is equivalent to 300 hours. Supposing 15 work hours per week, the total duration of this project is 20 weeks. This way, the beginning of this project would be September 19th, 2016, its ending would be February 3, 2017, and we would have 2 weeks between the end and the delivery so as to check all the project. The planning is divided in: Documentation, Project offer, Technology study, Analysis, Design, Implementation, Establishment, Evaluation, Planning.
- **Cost and profit**. The costs of the project are divided into:
  - **Staff cost**. This cost is associated to the employee's salary. The employee profile of this project is the Database Administrator. The Database Administrator's salary is approximately 2.213€ per month. Also, we have to add 661,7€ in taxes, so that the salary per month is 2.874,7€. As the project duration is 5 months, the money we have to pay this employee will be 14.373,5€.
  - **Equipment cost**. This cost is associated to the computers where the project will be established and the tools used to develop it. The computer has a price of 494€ and the tool used to develop the project is Cassandra, which is a free tool.
  - **Indirect cost**. This cost is associated to the equipment used to develop the project and other associated costs like the one of the place where the project will be developed. The equipment includes a laptop, a TV used as a monitor, a printer and Microsoft Office. We have just used this equipment for 20 weeks, so we have to decrease its price to 10% of its real price, which is 121,01€. Lastly, the place where the project will be developed has a price of 500€.

  After applying a risk and profit margin, the final cost is 19.360,64€.
- **Socio-economic impact**. This impact refers to the impact the project realization produces on the environment. That impact might be positive or negative, so I am going to study it in order to determine how the impact will be. There are two important impacts this project produces:
  - **Documentation**. As this project has a decision tree to help people to decide what NoSQL system they have to use, this help guide could be used by anyone who would like to know which NoSQL system to use or reaffirm which they have already choose. Therefore, I consider this is a positive impact.
  - **System chosen**. This impact is also positive because it provides some good features to the user, like giving a solution to the user necessities, always being available or getting used to environment easily.

Conclusion

Finally, this section is about the conclusions of this project. These conclusions are divided into:

- **Product conclusions**. Once the project is ended, I can say it has achieved all the goals I established at its beginning.
- **Process conclusions**. This project has caused me many problems that could have delayed the planning, but, fortunately, they were solved quickly. One of them was to obtain the data stored in LDAP, because I had to create some code programs to get that information from the UC3M website. Another one was that, sometimes, Cassandra did not let me login because there was an error.
- **Personal conclusions**. Personally, I am very proud of this project. All this time I have been working on it has made me learn a lot of things I did not know before, and I have been able to learn them on my own. I have also achieved the goal I established for myself when I started this project: to learn more about databases, which was my weak point during my degree course. The next subjects have helped me to finish this project: Files and Databases, Software Engineering, Software Development Projects Management, User Interfaces, Operating Systems, Computer Architecture, Operating Systems Design and Distributed Systems.

Apéndice III – Manual de instalación

Instalación de Cygwin64

Para instalar Cygwin, descargamos de la página de Oracle [43] la versión acorde con el Sistema Operativo donde vayamos a instalarlo. En mi caso, voy a instalarlo en un SO de 64 bits, por tanto, me he descargado la versión 64-bit.

Ejecutamos el archivo de instalación que acabamos de descargar y seguimos los pasos que nos indica el programa de instalación.

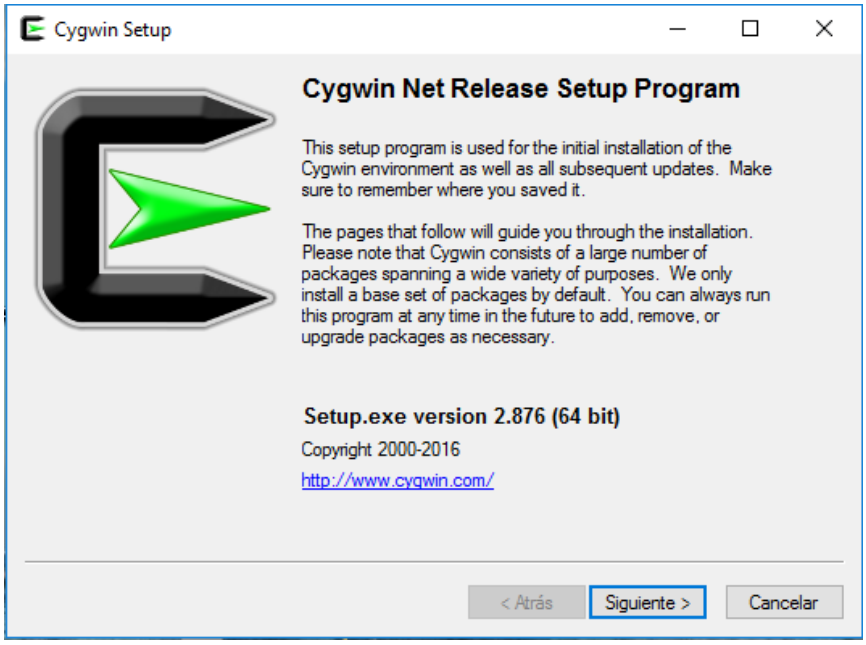

Ilustración 35: Instalar Cygwin - Paso 1

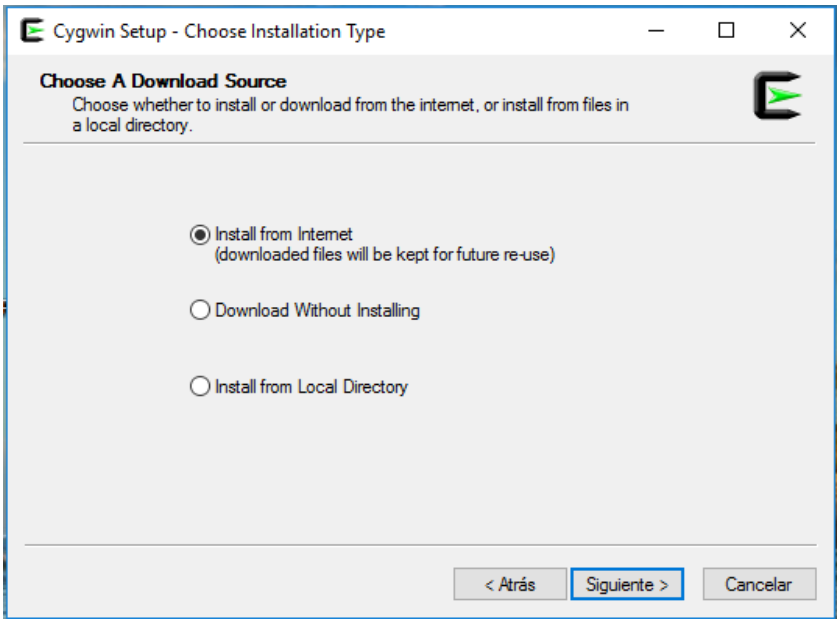

Ilustración 36: Instalar Cygwin - Paso 2

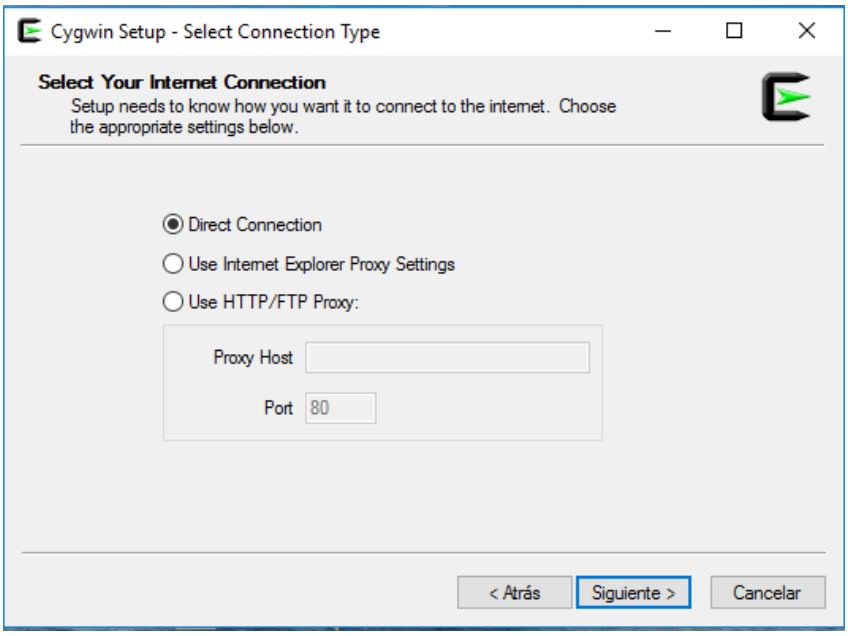

Ilustración 37: Instalar Cygwin - Paso 3

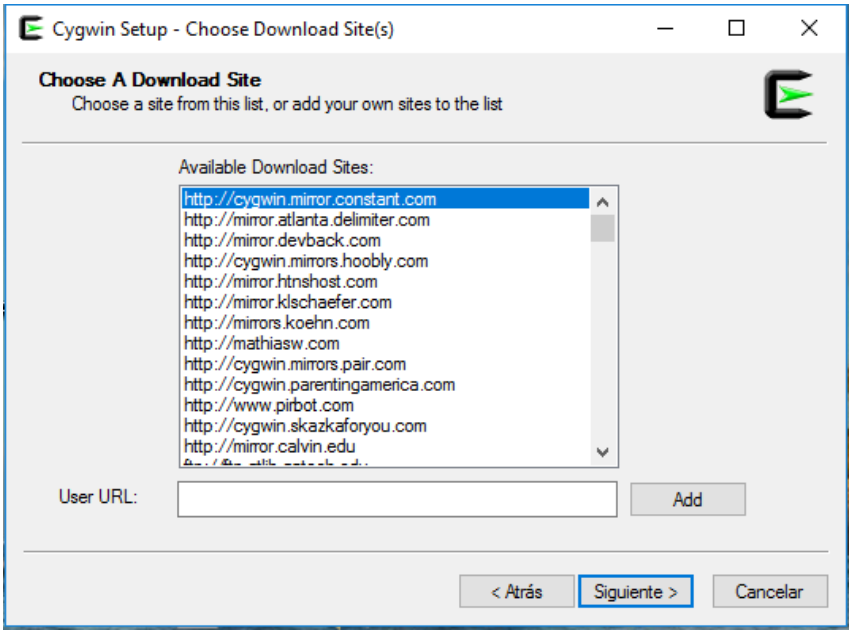

Ilustración 38: Instalar Cygwin - Paso 4

Para agregar la herramienta *curl*, seguimos los pasos indicados en [43].

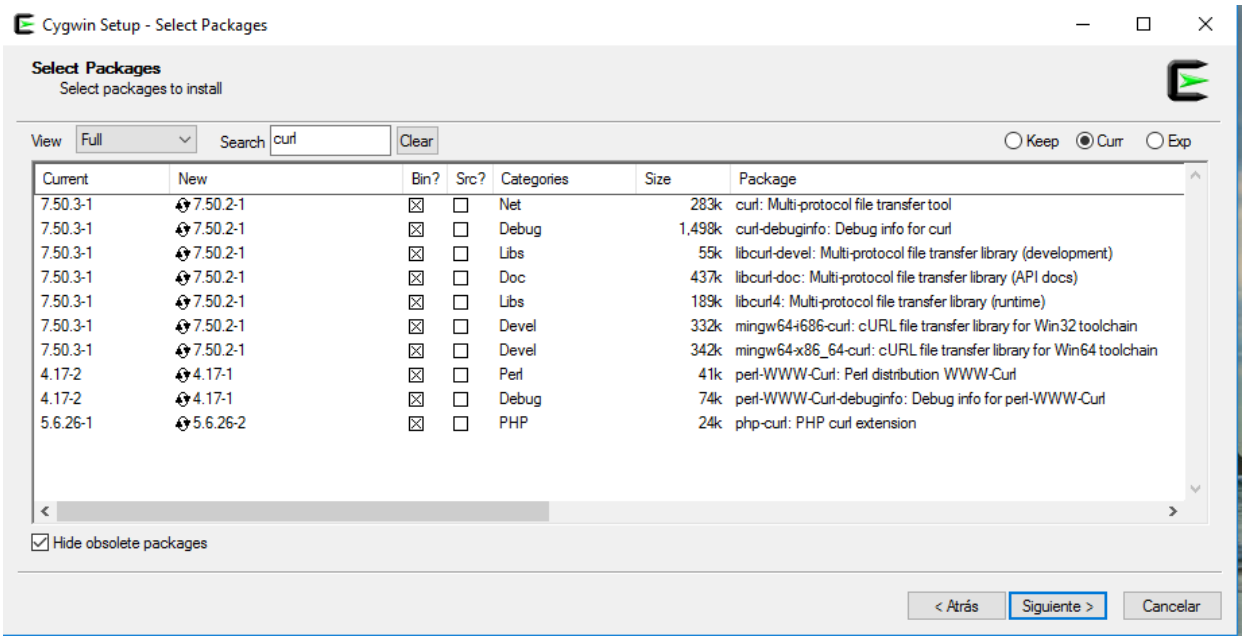

Ilustración 39: Instalar Cygwin - Paso 5

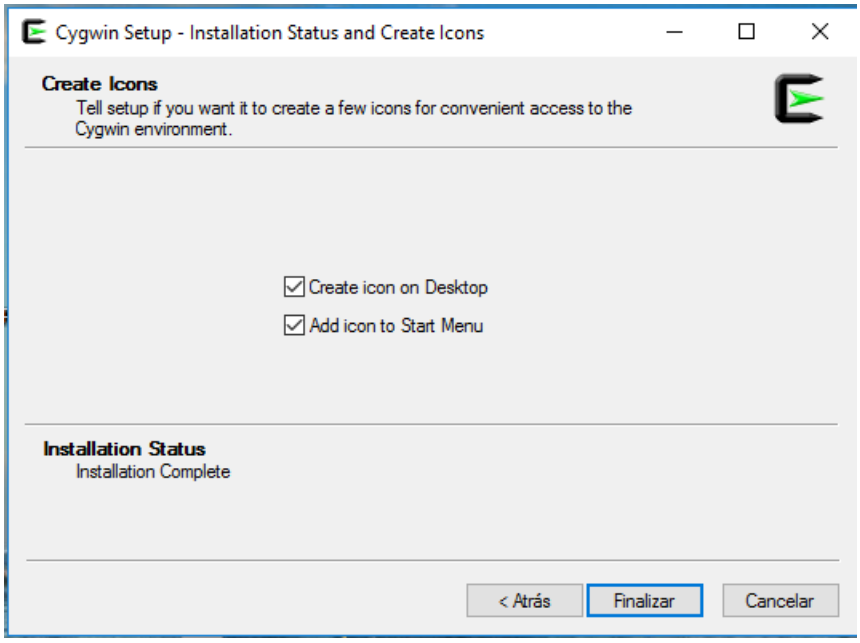

Ilustración 40: Instalar Cygwin - Paso 6

Instalación del entorno PHP

Para crear un entorno de ejecución de PHP, simplemente tenemos que descargarnos un archivo con extensión zip de [44] y descomprimirlo. El lugar donde lo descomprimamos será donde tendremos que ejecutar el código PHP.

Instalación de Oracle VM VirtualBox

Lo primero es acceder al apartado de Descargas de la página web de VirtualBox [45] y descargar el programa de instalación para el Sistema Operativo que elijamos. En este caso he elegido instalar el entorno virtual en un sistema Windows 10 Pro, y en un ordenador con las siguientes características:

- **Procesador**: Intel Core i7-5500U 2.4GHz.
- **RAM**: 8 GB.
- **Tipo de sistema**: Sistema operativo de 64 bits.

Una vez que tenemos el archivo de instalación, procedemos a ejecutarlo haciendo doble click en él (si hemos elegido Windows como sistema de instalación). Se nos abrirá un asistente de instalación, el cual debemos seguir paso a paso según nos indican. Cuando termine, ya tendremos instalado VirtualBox en nuestro sistema. En este caso he instalado la versión 5.0.26.

Creación de la máquina virtual

Si ya disponemos del programa VirtualBox instalado, procedemos a abrirlo (en caso contrario, mirar apartado *INSTALACIÓN DE ORACLE VM VIRTUALBOX*). Una vez abierto, nos saldrá una imagen como la siguiente.

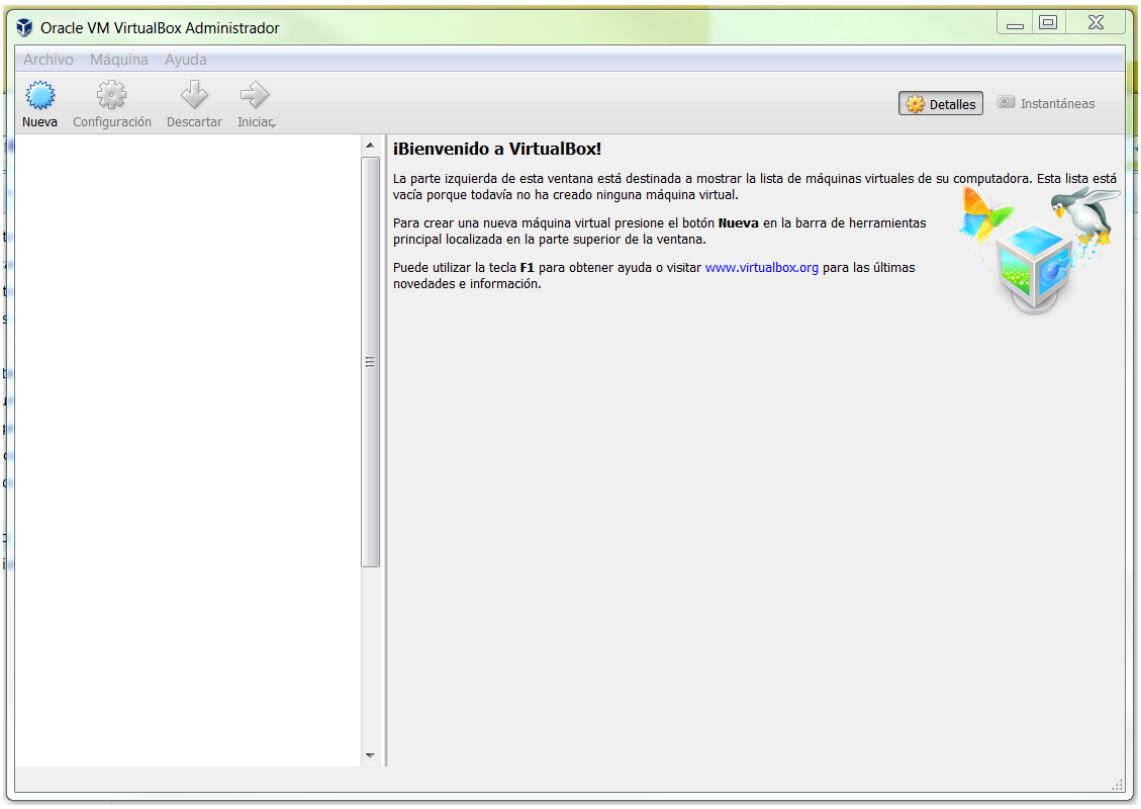

Ilustración 41: Ventana de inicio VirtualBox

Para crear una máquina virtual, pulsamos sobre Nueva, y nos aparecerá una ventana como la siguiente. Aquí elegimos el nombre que queramos darle a nuestra máquina virtual, el tipo de sistema sobre el que quieres montarla (Windows, Linux, Solaris, etc.), la versión (32 o 64 bits), la memoria que quieres asignarle (como se puede apreciar, el programa te muestra la memoria máxima del ordenador en MB, y en colores la memoria que puedes asignarle sin que ralentice el funcionamiento del mismo, es decir, en verde el tamaño que puedes asignarle sin que interfiera en el funcionamiento normal del ordenador, en naranja si interfiere un poco pero lo suficiente para hacer uso de él a la vez que está la máquina virtual abierta, y en rojo si interfiere tanto que impide su normal funcionamiento) y, por último, el disco duro que le vas a asignar a la máquina virtual, pudiendo elegir entre no asignarle ninguno, crear uno nuevo, o asignarle uno existente. Una vez elegido todo esto, pulsamos en Crear.

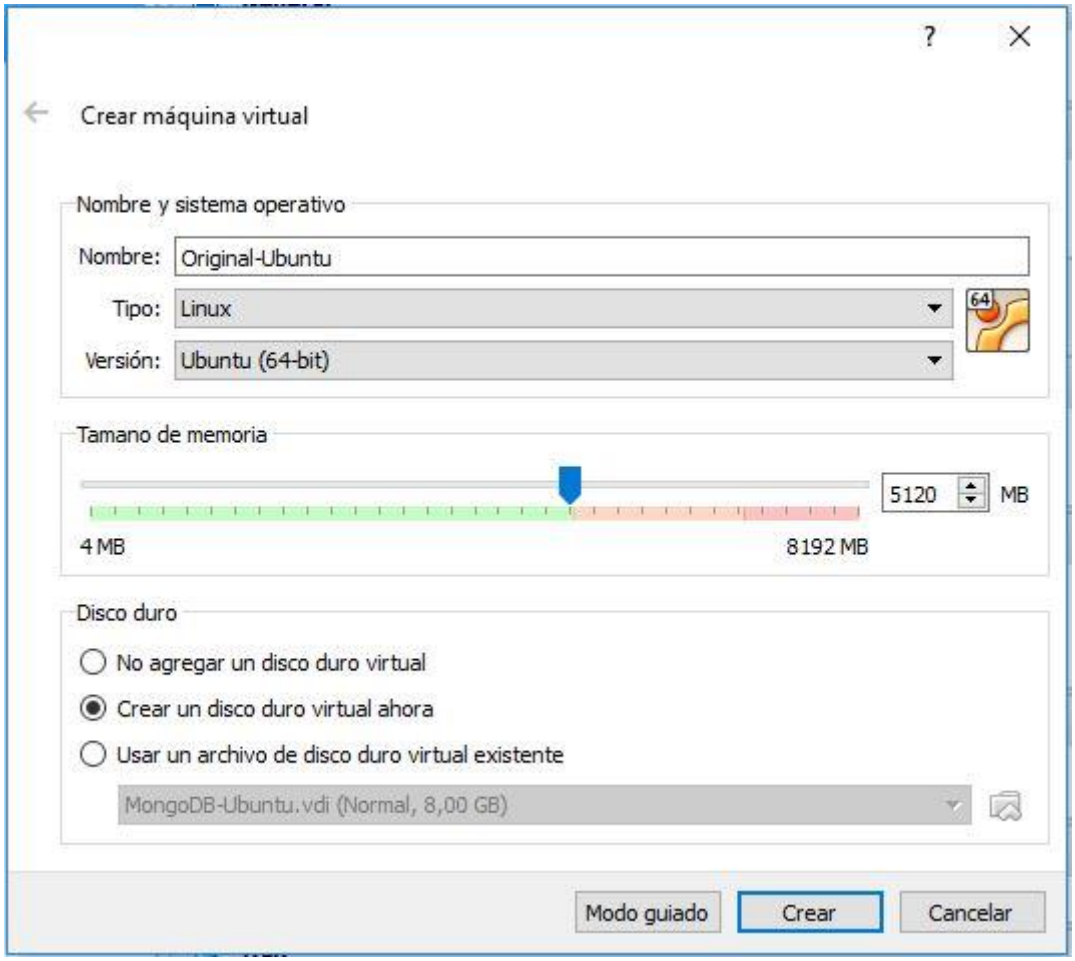

Ilustración 42: Crear máquina virtual

Al darle a Crear en el paso anterior, nos salta una ventana como la que se muestra a continuación sólo si decidimos crear un disco duro virtual nuevo. En ese caso, tendremos que elegir las características del mismo, como son el tamaño total que le queremos asignar (nuevamente el programa te muestra el tamaño total del disco duro del ordenador), el tipo de archivo de disco duro que queremos, entre una variedad de opciones, y si queremos que el disco duro vaya creciendo de manera dinámica, es decir, que según vaya necesitando vaya usando espacio hasta llegar al límite marcado antes, pero si no lo necesita ese espacio lo puede usar otro proceso, o que sea de tamaño fijo, es decir, que se reserve el tamaño elegido antes para la máquina virtual y ningún otro proceso pueda hacer uso de ese espacio (la primera opción es mejor de cara a almacenamiento, pero es más lenta que la segunda al tener que reservar y liberar espacio).

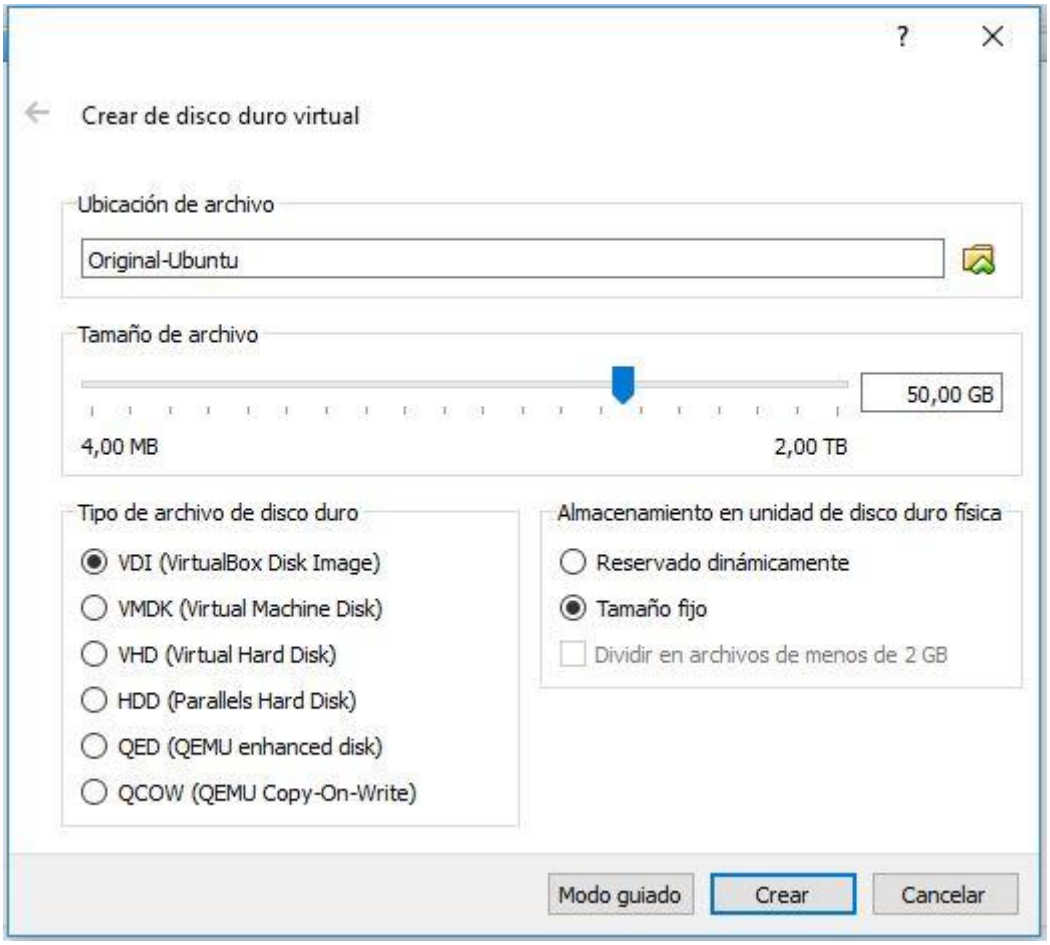

Ilustración 43: Crear disco duro virtual

Cuando terminemos ya tendremos la máquina virtual creada, la cual nos aparecerá en la parte izquierda. Ahora, pulsamos en ella y a continuación en Configuración. Necesitamos agregar un disco de instalación del sistema operativo que hayamos decidido montar en la máquina virtual para poder usarlo. Por tanto, en la parte izquierda, seleccionamos Almacenamiento, y luego en el disco con un + que aparece a la derecha de 'Controlador: IDE'. Al hacerlo, saldrá una ventana que nos preguntará si queremos dejarlo vacío o seleccionar un disco. Elegimos lo último y seleccionamos el archivo de instalación de Ubuntu. Este archivo lo encontramos en la sección Descargas de la página web de Ubuntu [46], seleccionando Ubuntu Desktop. En este caso he instalado la versión 16.04.1.

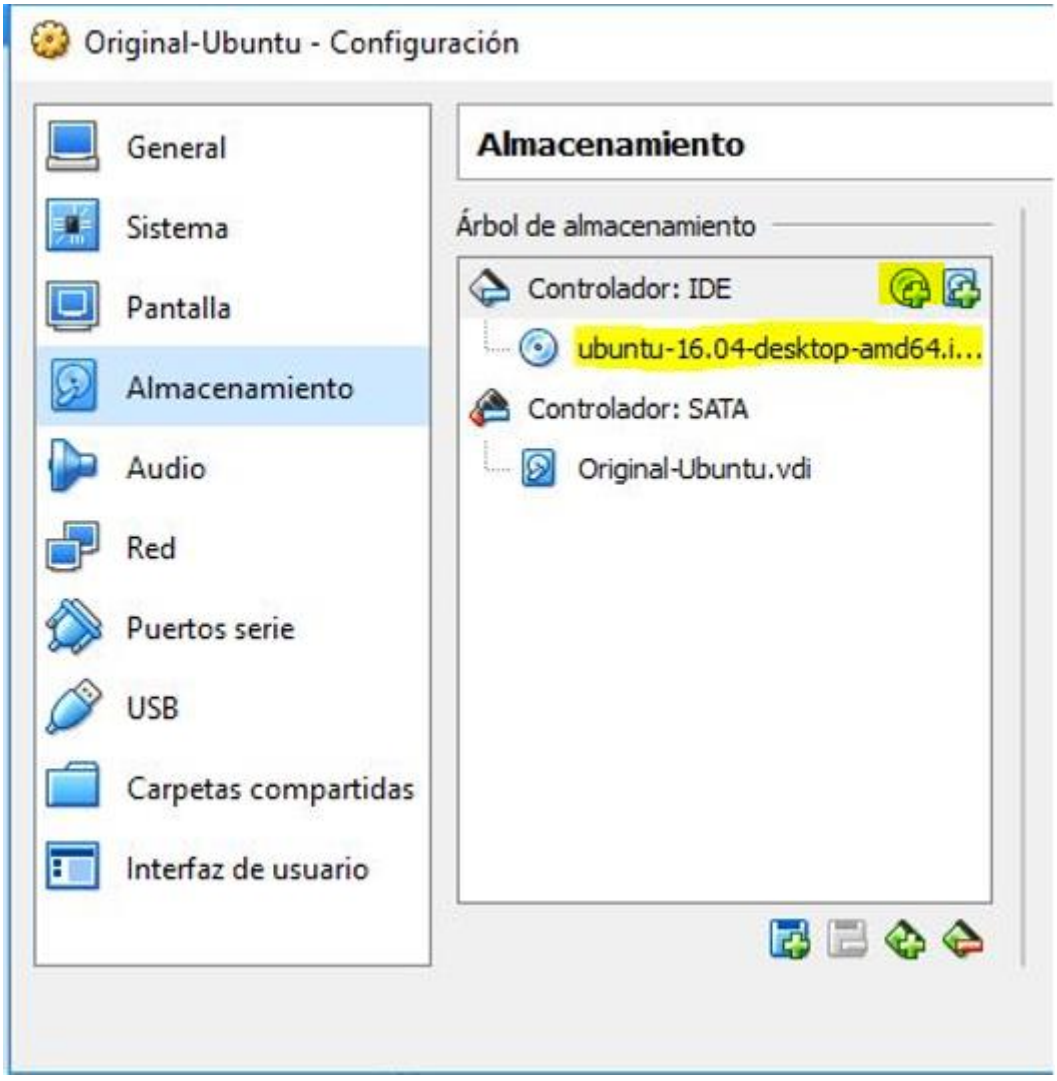

Ilustración 44: Asignar disco de instalación

Ahora ya sólo nos queda instalar el Sistema Operativo en la máquina virtual. Para ello, la seleccionamos y pulsamos en Iniciar. Tras el arranque, nos saldrá la pantalla de instalación de Ubuntu, como se muestra en la siguiente imagen. Pulsamos en Instalar Ubuntu y seguimos las instrucciones de instalación.

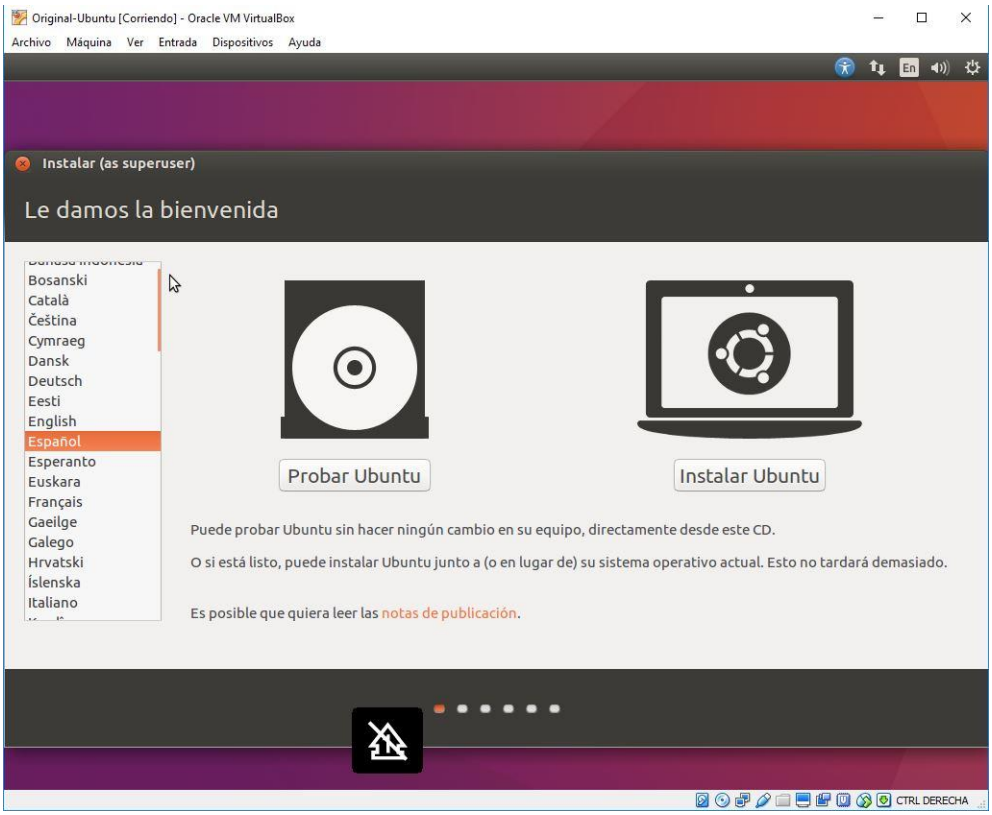

Ilustración 45: Instalar Ubuntu – Paso 1

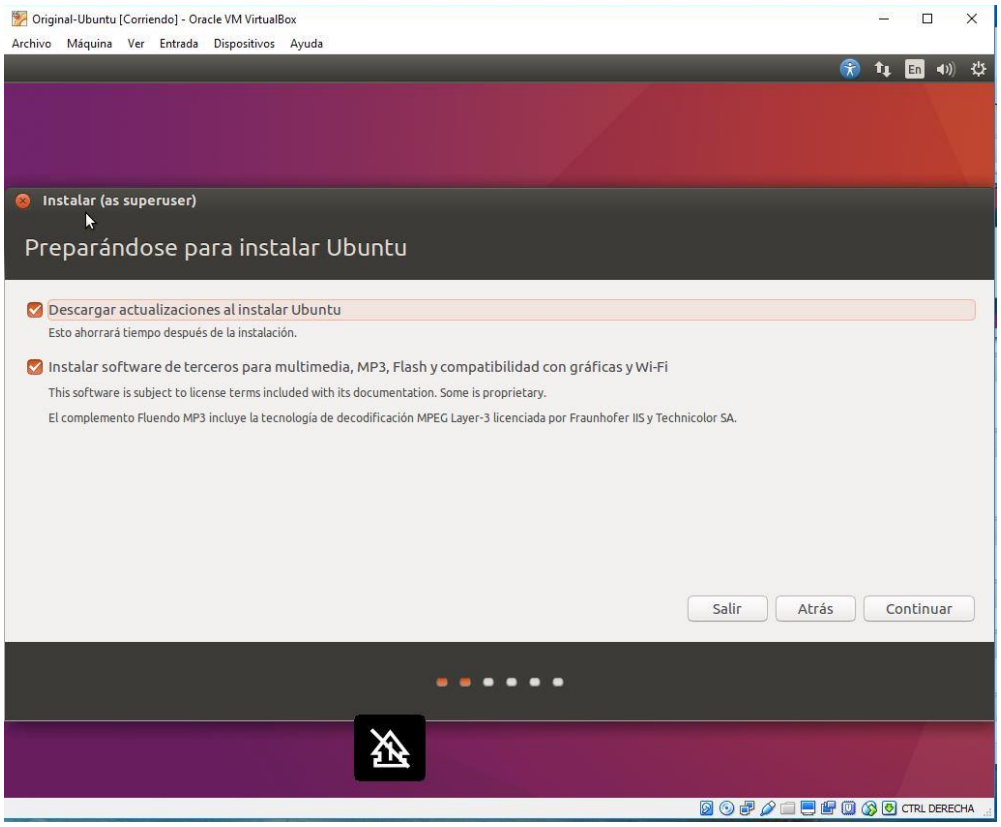

Ilustración 46: Instalar Ubuntu - Paso 2

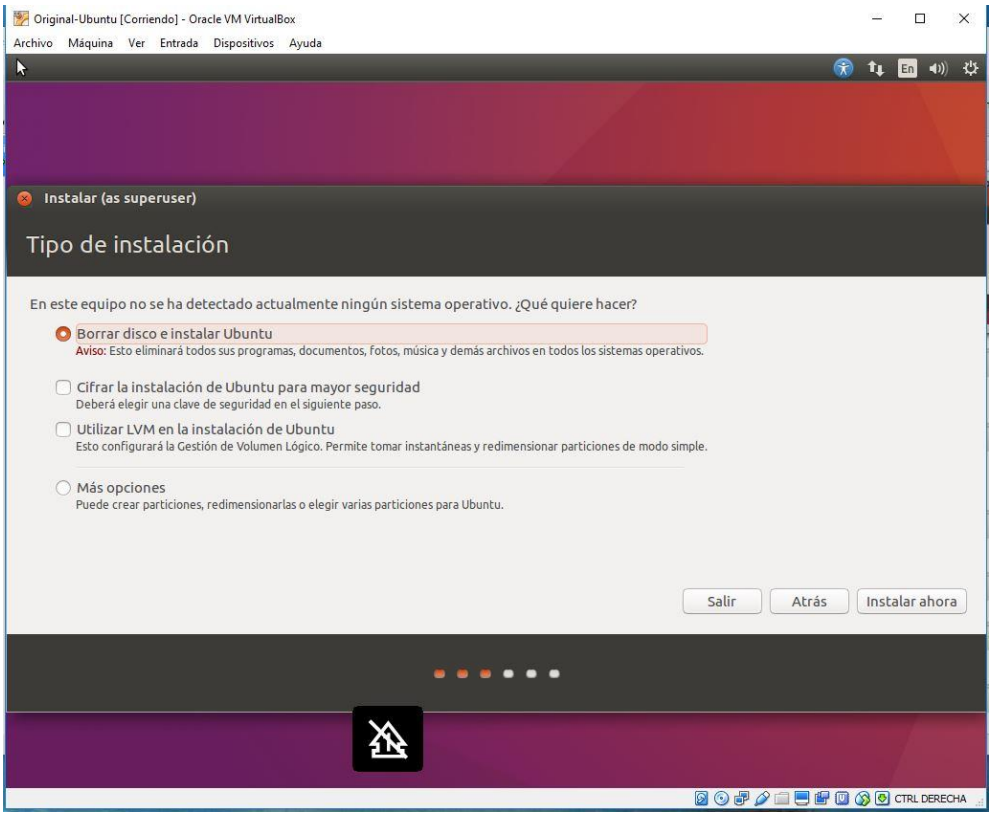

Ilustración 47: Instalar Ubuntu - Paso 3

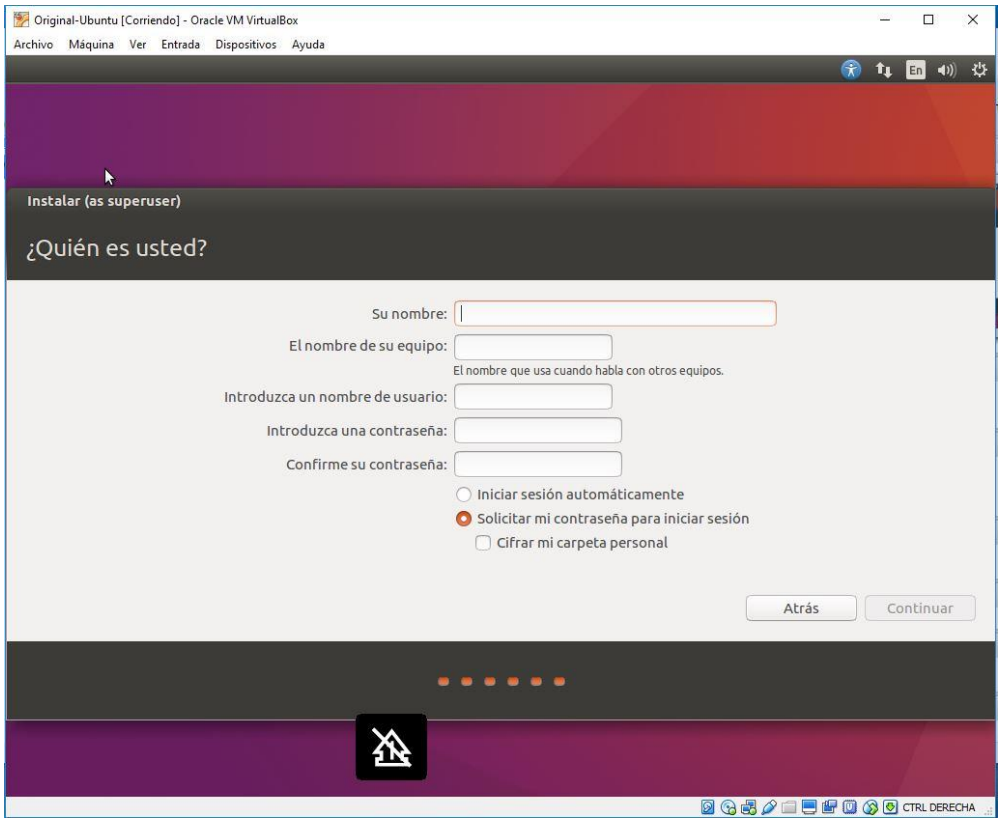

Ilustración 48: Instalar Ubuntu - Paso 4

Ya hemos creado nuestra máquina virtual. Ahora vamos a clonarla y guardar la máquina original, por si tuviéramos algún problema, poder volver a clonar la original en vez de tener que crearla. Para ello, hacemos click derecho en la máquina recién creada y elegimos la opción clonar. Nos saldrá una ventana como la siguiente, donde elegimos el nombre de la nueva máquina virtual, el tipo de clonación que queremos y lo que queremos clonar, el estado actual de la máquina o toda ella.

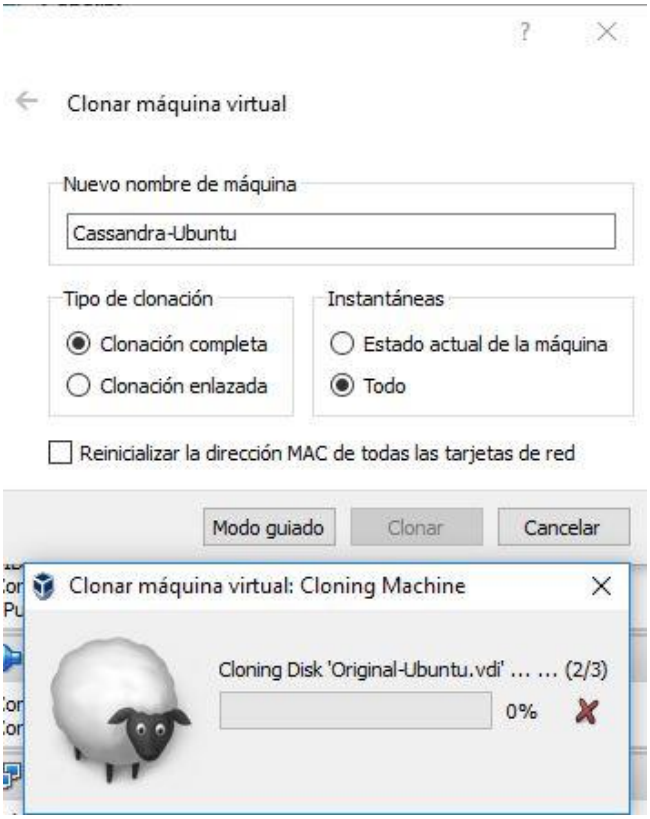

Ilustración 49: Clonar máquina virtual

Pasado un largo rato, ya tendremos nuestra máquina virtual clonada con las mismas características de las que dotamos a la original, y nos quedará algo como lo que se muestra a continuación.

Ilustración 50: Disposición final de máquinas virtuales

Instalación de Cassandra

Para instalar Cassandra, primero abrimos la máquina virtual en la que realizaremos la instalación (si no disponemos de dicha máquina virtual, mirar apartado *CREACIÓN DE LA MÁQUINA VIRTUAL*).

Una vez dentro de la máquina virtual nos encontramos el entorno de Ubuntu. Para realizar la instalación, he seguido detenidamente los pasos señalados en la web [47]. En las siguientes imágenes se muestran algunos de ellos y el resultado que he obtenido.

```
sergio@sergio-VirtualBox:~$ java -version
java version "1.8.0_101"
Java(TM) SE Runtime Environment (build 1.8.0_101-b13)
Java HotSpot(TM) 64-Bit Server VM (build 25.101-b13, mixed mode)
```

Ilustración 51: Instalar Cassandra - Paso 1

```
sergio@sergio-VirtualBox:~$ echo "deb http://www.apache.org/dist/cassandra/debia
n 37x main" | sudo tee -a /etc/apt/sources.list.d/cassandra.sources.list
```

Ilustración 52: Instalar Cassandra - Paso 2

```
sergio@sergio-VirtualBox:~$ sudo service cassandra status
● cassandra.service - LSB: distributed storage system for structured data
   Loaded: loaded (/etc/init.d/cassandra; bad; vendor preset: enabled)
   Active: active (running) since mié 2016-09-14 18:17:42 CEST; 9min ago
     Docs: man:systemd-sysv-generator(8)
   CGroup: /system.slice/cassandra.service
           └─5205 java -Xloggc:/var/log/cassandra/gc.log -ea -XX:+UseThreadPrior

sep 14 18:17:42 sergio-VirtualBox systemd[1]: Starting LSB: distributed storage
sep 14 18:17:42 sergio-VirtualBox systemd[1]: Started LSB: distributed storage s
```

Ilustración 53: Comprobar servicio Cassandra

En el punto 3 del manual dice que, debido a un bug, Cassandra no arranca automáticamente después de la instalación. Sin embargo, a mí sí me arrancó, por lo que no tuve que seguir los pasos que indican en ese punto y pasé directamente al 4.

```
sergio@sergio-VirtualBox:~$ sudo nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address    Load       Tokens       Owns (effective)  Host ID                               Rack
UN  127.0.0.1  102,98 KiB  256          100,0%            5d25b8c5-e2ad-44c8-9c16-53109cb2d850  rack1

sergio@sergio-VirtualBox:~$ cqlsh
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.7 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
cqlsh>
```

Ilustración 54: Ejecutar Cassandra

Por último, para probar que Cassandra se había instalado correctamente, realicé algunos ejemplos sencillos que encontré en la web [48].

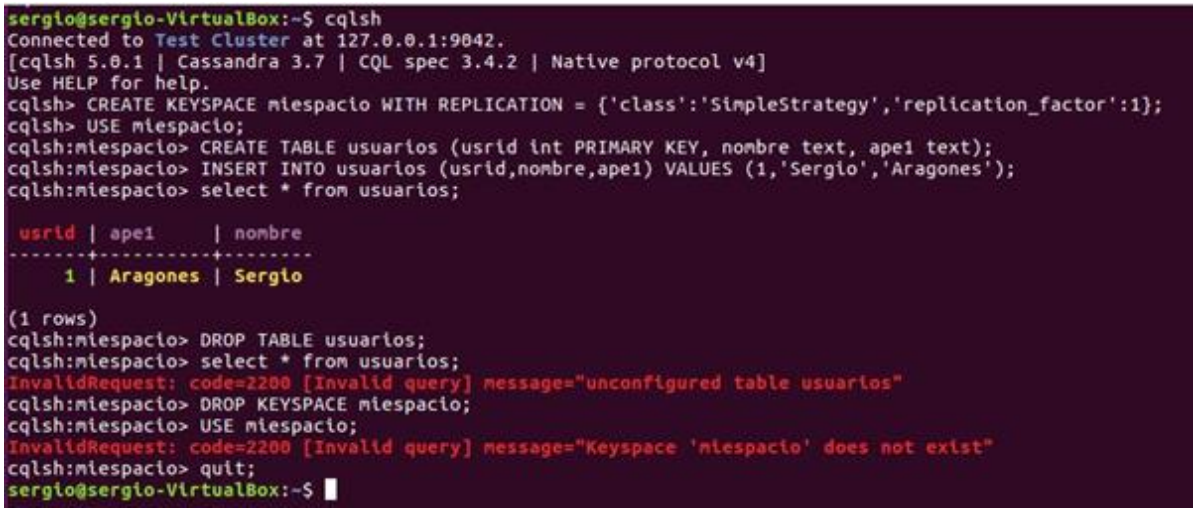

Ilustración 55: Pruebas en Cassandra

Para este proyecto, la versión de Cassandra que estoy utilizando es la 3.7, que puede verse en las imágenes anteriores al entrar con el comando *cqlsh*.

Apéndice IV - Descripción del código

codigoHTML.txt

Este archivo se obtiene de la ejecución del comando *curl* sobre la página del Personal del Departamento de Informática de la UC3M. En este archivo se guarda el código HTML descargado de dicha página. La información que nos interesa de este archivo son las URLs de cada uno de los miembros del personal, para luego acceder a ellas y descargar el código de esas páginas. Como se puede ver en la web [49], el personal está agrupado en distintas categorías. En el código, cada categoría está definida dentro de un elemento llamado *page-header*, y a continuación de él se encuentran las URLs de los miembros de esa categoría. En la siguiente imagen se muestra gráficamente:

```
<div class="page-header">
                    <h2 itemprop="name">
                                        Catedráticos                    </h2>

                </div>

<p>
<br><p><a href="/./es/component/comprofiler/userprofile/aedo">Aedo Cuevas, Ignacio</a><br><a href="/./es/component/comprofiler/userprofile/dborrajo">Borrajo Millán, Daniel</a><br><a
href="/./es/component/comprofiler/userprofile/jcarrete">Carretero Pérez, Jesús</a><br><a href="/./es/component/comprofiler/userprofile/amescua">de Amescua Seco, Antonio</a><br><a
href="/./es/component/comprofiler/userprofile/pdp">Díaz Pérez, Paloma</a><br><a href="/./es/component/comprofiler/userprofile/fgcarbal">García Carballeira, Félix</a><br><a
href="/./es/component/comprofiler/userprofile/isasi">Isasi Viñuela, Pedro</a><br><a href="/./es/component/comprofiler/userprofile/llorens">Llorens Morillo, Juan Bautista</a><br><a
href="/./es/component/comprofiler/userprofile/molina">Molina López, José Manuel</a><br><a href="/./es/component/comprofiler/userprofile/arturo">Ribagorda Garnacho, Arturo</a><br></p>
</p>
```

Ilustración 56: Trozo de código del archivo códigoHTML.txt

codigo1.c

Este código se ha desarrollado para extraer todas las URLs del archivo *códigoHTML.txt* y escribirlas en el archivo *salida.sh*. Para ello, lo primero es abrir ambos archivos.

Lo siguiente es, dentro de un bucle, recorrer el archivo entero carácter a carácter hasta que encontremos la cadena de caracteres *page-header*, que es donde hallaremos las URLs. Cuando la encontremos (*ILUSTRACIÓN 57: TROZO 1 DE CÓDIGO DEL ARCHIVO CODIGO1.C*), entramos en otro bucle para encontrar la cadena de caracteres *href*, que es lo que habrá justo antes de cada URL.

```
if(c=='p'){
        read(fd1,&c2,10);
        if(strcmp(c2,"age-header")==0){
                while(1){
```

Ilustración 57: Trozo 1 de código del archivo codigo1.c

Cuando encontremos la cadena *href* (*ILUSTRACIÓN 58: TROZO 2 DE CÓDIGO DEL ARCHIVO CODIGO1.C*), leemos los siguientes 5 caracteres, que son los correspondientes a *=”/./*, para que lo siguiente que leamos ya sea información válida.

```
if(c=='h'){
        read(fd1,&a,1);read(fd1,&b,1);read(fd1,&c,1);
       if(a=='r' && b=='e' && c=='f'){
```

Ilustración 58: Trozo 2 de código del archivo codigo1.c

Ahora inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *curl http://www.inf.uc3m. '* y a continuación vamos introduciendo los caracteres que leamos del archivo para completar la URL. Sabremos que hemos llegado al final de la URL cuando encontremos unas comillas dobles (“). Entonces, añadiremos a *salida* la cadena ' > salidas/s ', la variable *n* (un número que se irá incrementando por cada URL que encontremos) y, finalmente, la cadena '.txt' y un salto de línea (\n). Justo después escribiremos la variable *salida* en el archivo *salida.sh*.

Después, volvemos a buscar la cadena *href* para encontrar al siguiente miembro. Haremos esto hasta que encontremos la cadena de caracteres *!--*. Entonces habremos llegado al final de la categoría, y tendremos que buscar otra vez la cadena *page-header* para encontrar la siguiente.

Cuando no haya más caracteres que leer, cerramos los dos archivos y terminamos el programa.

salida.sh

En este archivo se guarda, por cada miembro del Dpto. de Informática, una cadena de caracteres con el comando *curl*, la URL de la que se quiere obtener el código y el archivo de salida donde se volcará.

```
curl http://www.inf.uc3m.es/component/comprofiler/userprofile/aedo > salidas/s1.txt
curl http://www.inf.uc3m.es/component/comprofiler/userprofile/dborrajo > salidas/s2.txt
curl http://www.inf.uc3m.es/component/comprofiler/userprofile/jcarrete > salidas/s3.txt
curl http://www.inf.uc3m.es/component/comprofiler/userprofile/amescua > salidas/s4.txt
```

Ilustración 59: Trozo de código del archivo salida.sh

Tras la ejecución de este script, tendremos una carpeta llamada *salidas* con un archivo por cada uno de los miembros del Dpto. de Informática. Este archivo contendrá toda la información que se quiere obtener del usuario.

codigo2.c

Este código se ha desarrollado porque, en el archivo con la información del usuario, el correo viene codificado. Para decodificarlo, hace falta escribirlo en un archivo que posteriormente se decodificará usando PHP. Por tanto, este código tiene como objetivo escribir en un archivo los correos codificados de todos los miembros del Dpto. de Informática. Estos correos los sacará de los archivos creados tras la ejecución del script *salida.sh*.

El siguiente procedimiento se realizará para cada uno de los archivos de la carpeta salidas. Lo primero es abrir el archivo con la información del usuario y recorrerlo entero carácter a carácter hasta que encontremos la cadena de caracteres *path*, que es donde encontraremos el correo codificado. Cuando la encontremos (*ILUSTRACIÓN 60: TROZO 1 DE CÓDIGO DEL ARCHIVO CODIGO2.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, y le añadimos la cadena ' *echo html_entity_decode(“* '.

```
if(c=='p'){
        read(fd1,&a,1);read(fd1,&b,1);read(fd1,&c,1);
        if(a=='a' && b=='t' && c=='h'){//path
                memset(salida, '\0', sizeof(salida));
                strcpy(salida,"echo html_entity_decode(");
```

Ilustración 60: Trozo 1 de código del archivo codigo2.c

Ahora leemos caracteres hasta encontrar el primer carácter codificado. El correo viene codificado con códigos HTML. Dichos códigos tienen el formato *&#x*, siendo *x* un número cualquiera [50]. A continuación, vamos introduciendo los códigos HTML en la variable *salida*, evitando meter caracteres como espacios, comillas simples (') o sumas (+). Sabremos que hemos llegado al final del correo cuando encontremos un salto de línea (\n). Entonces, añadiremos a *salida* la cadena ' *“).”*\n*”;* '. Por último, cerramos el archivo de entrada y escribimos la variable en el archivo *codigoPHP.php*.

codigoPHP.php

En este archivo se guarda, por cada miembro del Dpto. de Informática, una cadena de caracteres con el comando *html_entity_decode* y los códigos HTML a decodificar.

```
<?php
echo html_entity_decode("&#97;&#101;d&#111;&#64;&#105;&#97;&#46;&#117;c3m&#46;&#101;s")."\n";
echo html_entity_decode("db&#111;rr&#97;j&#111;&#64;&#105;&#97;&#46;&#117;c3m&#46;&#101;s")."\n";
echo html_entity_decode("jc&#97;rr&#101;t&#101;&#64;&#105;nf&#46;&#117;c3m&#46;&#101;s")."\n";
```

Ilustración 61: Trozo de código del archivo codigoPHP.php

correos.txt

Este archivo se obtiene de la ejecución del archivo *codigoPHP.php*, y contiene los correos decodificados de todos los miembros del Dpto. de Informática.

```
aedo@ia.uc3m.es
dborrajo@ia.uc3m.es
jcarrete@inf.uc3m.es
amescua@inf.uc3m.es
pdp@inf.uc3m.es
```

Ilustración 62: Trozo de código del archivo correos.txt

codigo3.c

Este código se ha desarrollado para extraer la información de los miembros del Dpto. de Informática almacenada en los archivos de la carpeta *salidas* que se crean tras la ejecución de *salida.sh*. Como en estos archivos el correo viene codificado, esta información tiene que extraerse el archivo *correos.txt* creado tras la ejecución de *codigoPHP.php*. Por tanto, lo primero es abrir el archivo *correos.txt* y luego el primer archivo de la carpeta *salidas*.

Lo primero que vamos a almacenar es el correo. Para ello, inicializamos la variable donde se guardarán los datos, llamada *sal*, le añadimos la cadena ' *Email: '* y a continuación vamos introduciendo los caracteres que leamos del archivo para completar el email. Sabremos que hemos llegado al final del correo cuando encontremos un salto de línea (\n). Entonces, salimos del bucle y nos disponemos a leer el resto de información del usuario.

Para el resto de información, abrimos el archivo de la carpeta *salidas*. Como tenemos varios campos que almacenar, tendremos que buscarlos de uno en uno.

Para almacenar el Nombre, tenemos que buscar en el código la cadena *Nomb*. Cuando la encontremos (*ILUSTRACIÓN 63: TROZO 1 DE CÓDIGO DEL ARCHIVO CODIGO3.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *Nombre: '* y a continuación leemos hasta encontrar dos puntos (:) y luego los siguientes 48 caracteres, para llegar al inicio del nombre. Al hacerlo, entramos en un bucle y vamos guardando la información en la variable *salida*. Sabremos que hemos llegado al final del nombre cuando encontremos un menor (<). Entonces, escribimos la variable *salida* en el archivo *datosLDAP.txt*.

```
/*  NOMBRE  */
else if(c=='N'){
        read(fd1,&a,1);read(fd1,&b,1);read(fd1,&c,1);
        if(a=='o' && b=='m' && c=='b'){
```

Ilustración 63: Trozo 1 de código del archivo codigo3.c

Para almacenar los Apellidos, tenemos que buscar en el código la cadena *Apel*. Cuando la encontremos (*ILUSTRACIÓN 64: TROZO 2 DE CÓDIGO DEL ARCHIVO CODIGO3.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *Apellidos: '* y a continuación leemos hasta encontrar dos puntos (:) y luego los siguientes 48 caracteres, para llegar al inicio de los apellidos. Al hacerlo, entramos en un bucle y vamos guardando la información en la variable *salida*. Sabremos que hemos llegado al final de los apellidos cuando encontremos un menor (<). Entonces, escribimos la variable *salida* en el archivo *datosLDAP.txt*.

```
/*  APELLIDOS  */
else if(c=='A'){
        read(fd1,&a,1);read(fd1,&b,1);read(fd1,&c,1);
        if(a=='p' && b=='e' && c=='l'){
```

Ilustración 64: Trozo 2 de código del archivo codigo3.c

Para almacenar el Teléfono, tenemos que buscar en el código la cadena *Tel&*. Cuando la encontremos (*ILUSTRACIÓN 65: TROZO 3 DE CÓDIGO DEL ARCHIVO CODIGO3.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *Telefono: '* y a continuación leemos hasta encontrar dos puntos (:) y luego los siguientes 48 caracteres, para llegar al inicio del teléfono. Al hacerlo, entramos en un bucle y vamos guardando la información en la variable *salida*. Sabremos que hemos llegado al final del teléfono cuando encontremos un menor (<). Entonces, escribimos la variable *salida* en el archivo *datosLDAP.txt*.

```
/*  TELEFONO  */
else if(c=='T'){
        read(fd1,&a,1);read(fd1,&b,1);read(fd1,&c,1);
        if(a=='e' && b=='l' && c=='&'){
```

Ilustración 65: Trozo 3 de código del archivo codigo3.c

Para almacenar el Campus, tenemos que buscar en el código la cadena *Camp*. Cuando la encontremos (*ILUSTRACIÓN 66: TROZO 4 DE CÓDIGO DEL ARCHIVO CODIGO3.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *Campus: '* y a continuación leemos hasta encontrar dos puntos (:) y luego los siguientes 48 caracteres, para llegar al inicio del campus. Al hacerlo, entramos en un bucle y vamos guardando la información en la variable *salida*. Sabremos que hemos llegado al final del campus cuando encontremos un menor (<). Entonces, escribimos la variable *salida* en el archivo *datosLDAP.txt*.

```
else if(c=='C'){
        read(fd1,&a,1);read(fd1,&b,1);read(fd1,&c,1);
        /*  CAMPUS  */
        if(a=='a' && b=='m' && c=='p'){
```

Ilustración 66: Trozo 4 de código del archivo codigo3.c

Para almacenar la Categoría, tenemos que buscar en el código la cadena *Cate*. Cuando la encontremos (*ILUSTRACIÓN 67: TROZO 5 DE CÓDIGO DEL ARCHIVO CODIGO3.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *Categoria: '* y a continuación leemos hasta encontrar dos puntos (:) y luego los siguientes 48 caracteres, para llegar al inicio de la categoría. Al hacerlo, entramos en un bucle y vamos guardando la información en la variable *salida*. Sabremos que hemos llegado al final de la categoría cuando encontremos un menor (<). Entonces, escribimos la variable *salida* en el archivo *datosLDAP.txt*.

```
/*  CATEGORIA  */
else if(a=='a' && b=='t' && c=='e'){
```

Ilustración 67: Trozo 5 de código del archivo codigo3.c

Para almacenar el Área de Conocimiento, tenemos que buscar en el código la cadena Cono. Cuando la encontremos (*ILUSTRACIÓN 68: TROZO 6 DE CÓDIGO DEL ARCHIVO CODIGO3.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *Area de Conocimiento: '* y a continuación leemos hasta encontrar dos puntos (:) y luego los siguientes 48 caracteres, para llegar al inicio del área de conocimiento. Al hacerlo, entramos en un bucle y vamos guardando la información en la variable *salida*. Sabremos que hemos llegado al final del área de conocimiento cuando encontremos un menor (<). Entonces, escribimos la variable *salida* en el archivo *datosLDAP.txt*.

```
/*  AREA DE CONOCIMIENTO  */
else if(a=='o' && b=='n' && c=='o'){
```

Ilustración 68: Trozo 6 de código del archivo codigo3.c

Para almacenar el Edificio, tenemos que buscar en el código la cadena *Edif*. Cuando la encontremos (*ILUSTRACIÓN 69: TROZO 7 DE CÓDIGO DEL ARCHIVO CODIGO3.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *Edificio: '* y a continuación leemos hasta encontrar dos puntos (:) y luego los siguientes 48 caracteres, para llegar al inicio del edificio. Al hacerlo, entramos en un bucle y vamos guardando la información en la variable *salida*. Sabremos que hemos llegado al final del edificio cuando encontremos un menor (<). Entonces, escribimos la variable *salida* en el archivo *datosLDAP.txt*.

```
/*  EDIFICIO  */
else if(c=='E'){
        read(fd1,&a,1);read(fd1,&b,1);read(fd1,&c,1);
        if(a=='d' && b=='i' && c=='f'){
```

Ilustración 69: Trozo 7 de código del archivo codigo3.c

Para almacenar el Despacho, tenemos que buscar en el código la cadena *Desp*. Cuando la encontremos (*ILUSTRACIÓN 70: TROZO 8 DE CÓDIGO DEL ARCHIVO CODIGO3.C*), inicializamos la variable donde se guardarán los datos, llamada *salida*, le añadimos la cadena ' *Despacho: '* y a continuación leemos hasta encontrar dos puntos (:) y luego los siguientes 48 caracteres, para llegar al inicio del despacho. Al hacerlo, entramos en un bucle y vamos guardando la información en la variable *salida*. Sabremos que hemos llegado al final del despacho cuando encontremos un menor (<). Entonces, escribimos la variable *salida* en el archivo *datosLDAP.txt*.

```
/*  DESPACHO  */
else if(c=='D'){
        read(fd1,&a,1);read(fd1,&b,1);read(fd1,&c,1);
        if(a=='e' && b=='s' && c=='p'){
```

Ilustración 70: Trozo 8 de código del archivo codigo3.c

Cuando tengamos toda la información almacenada, buscamos la cadena *L1">*, que está después del último campo que queremos almacenar, para terminar la ejecución y no tener que leer el resto del archivo innecesariamente.

Por último, realizaremos estos mismos pasos con el resto de los archivos de la carpeta *salidas* para recoger la información de todos los usuarios.

datosLDAP.txt

Este archivo se obtiene de la ejecución del archivo *codigo3.c*, y contiene la información de todos los miembros del Dpto. de Informática.

```
Nombre:Ignacio
Apellidos:Aedo Cuevas
Email:aedo@ia.uc3m.es
Telefono:916249490
Campus:Leganés
Edificio:Sabatini
Despacho:2.2.A.19
Categoria:Catedrático
Area de Conocimiento:Ciencia de la Computación e Inteligencia Artificial

Nombre:Daniel
Apellidos:Borrajo Millán
Email:dborrajo@ia.uc3m.es
Telefono:916249459
Campus:Leganés
Edificio:Sabatini
Despacho:2.1.B.09
Categoria:Catedrático
Area de Conocimiento:Ciencia de la Computación e Inteligencia Artificial
```

Ilustración 71: Trozo de código del archivo datosLDAP.txt

codigo3_toJSON.c

Este código se ha desarrollado para guardar la información de los miembros del departamento en un formato apto para que la página web JSON Generator [36] lo pueda convertir a formato JSON.

La funcionalidad es exactamente la misma que la del código original (*codigo3.c*).

datosLDAP_JSON.txt

Este archivo se obtiene de la ejecución del archivo *codigo3_toJSON.c*, y contiene la información de todos los miembros del Dpto. de Informática en un formato apto para la página web JSON Generator [36].

```
[
{
nombre:'Ignacio',
apellidos:'Aedo Cuevas',
email:'aedo@ia.uc3m.es',
telefono:'916249490',
campus:'Leganés',
edificio:'Sabatini',
despacho:'2.2.A.19',
categoria:'Catedrático',
area_de_conocimiento:'Ciencia de la Computación e Inteligencia Artificial'
},
{
nombre:'Daniel',
apellidos:'Borrajo Millán',
email:'dborrajo@ia.uc3m.es',
telefono:'916249459',
campus:'Leganés',
edificio:'Sabatini',
despacho:'2.1.B.09',
categoria:'Catedrático',
area_de_conocimiento:'Ciencia de la Computación e Inteligencia Artificial'
},
```

Ilustración 72: Trozo de código del archivo datosLDAP_toJSON.txt

generated.json

Este archivo se obtiene de la transformación del archivo *datosLDAP_toJSON.txt* a formato JSON a través de la página web JSON Generator [36].


```
[
  {
    "nombre": "Ignacio",
    "apellidos": "Aedo Cuevas",
    "email": "aedo@ia.uc3m.es",
    "telefono": 916249490,
    "campus": "Leganés",
    "edificio": "Sabatini",
    "despacho": "2.2.A.19",
    "categoria": "Catedrático",
    "area_de_conocimiento": "Ciencia de la Computación e Inteligencia Artificial"
  },
  {
    "nombre": "Daniel",
    "apellidos": "Borrajo Millán",
    "email": "dborrajo@ia.uc3m.es",
    "telefono": 916249459,
    "campus": "Leganés",
    "edificio": "Sabatini",
    "despacho": "2.1.B.09",
    "categoria": "Catedrático",
    "area_de_conocimiento": "Ciencia de la Computación e Inteligencia Artificial"
  },
```


Ilustración 73: Trozo de código del archivo generated.json

result.csv

Este archivo se obtiene de la transformación del archivo *generated.json* a formato CSV a través de la página web Konklone [37].


```
nombre,apellidos,email,telefono,campus,edificio,despacho,categoria,area_de_conocimiento
Ignacio,Aedo Cuevas,aedo@ia.uc3m.es,916249490,Leganés,Sabatini,2.2.A.19,Catedrático,Ciencia de la Computación e Inteligencia Artificial
Daniel,Borrajo Millán,dborrajo@ia.uc3m.es,916249459,Leganés,Sabatini,2.1.B.09,Catedrático,Ciencia de la Computación e Inteligencia Artificial
Jesús,Carretero Pérez,jcarrete@inf.uc3m.es,916249458,Leganés,Sabatini,2.2.A25,Catedrático,Arquitectura y Tecnología de Computadores
```


Ilustración 74: Trozo de código del archivo result.csv

Apéndice V - Obtención de los datos

PASO 1

Una vez instalado Cygwin64 (ver *INSTALACIÓN DE CYGWIN64*) con la herramienta *curl*, ejecutamos el siguiente comando: *curl http://www.inf.uc3m.es/personal > codigoHTML.txt*. El comando *curl* [51] descarga el contenido HTML de la página indicada, y el comando > lo vuelca en el archivo especificado después.

```
Sergio@Sergio ~
$ curl http://www.inf.uc3m.es/personal > codigoHTML.txt
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 58120    0 58120    0     0  86889      0 --:--:-- --:--:-- --:--:-- 99862

Sergio@Sergio ~
$
```

Ilustración 75: Descarga del código HTML de la web del Personal del Departamento de Informática

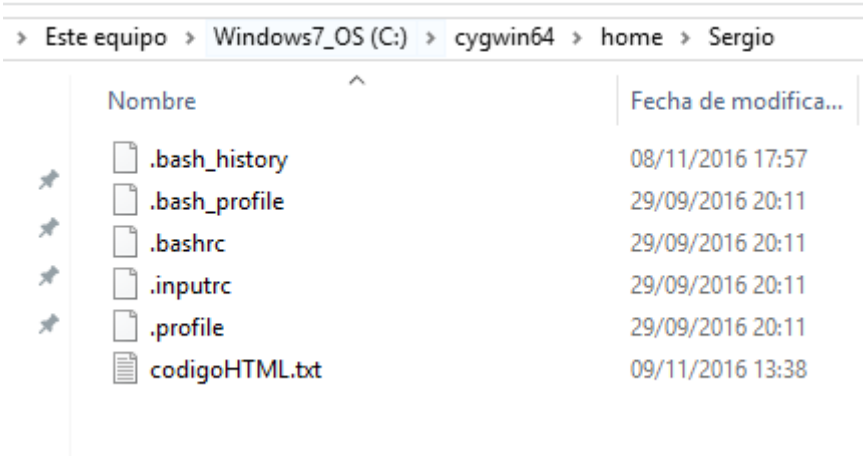

Ilustración 76: Distribución del directorio tras la descarga del código HTML

PASO 2

Ejecutamos *codigo1.c*. Este código busca en el archivo *codigoHTML.txt*, recoge las URL de los miembros del Departamento de Informática y las guarda en el archivo *salida.sh*.

```
sergio@sergio-VirtualBox: ~/Escritorio/codigo
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

sergio@sergio-VirtualBox:~$ cd Escritorio/codigo/
sergio@sergio-VirtualBox:~/Escritorio/codigo$ gcc -w codigo1.c -o codigo1.o
sergio@sergio-VirtualBox:~/Escritorio/codigo$ ./codigo1.o
sergio@sergio-VirtualBox:~/Escritorio/codigo$
```

Ilustración 77: Ejecución de código1

PASO 3

Copiamos el archivo *salida.sh* recién obtenido a la ruta C:/*cygwin64/home/Sergio* y creamos una carpeta llamada *salidas*, donde se guardarán los archivos con el código HTML de cada miembro.

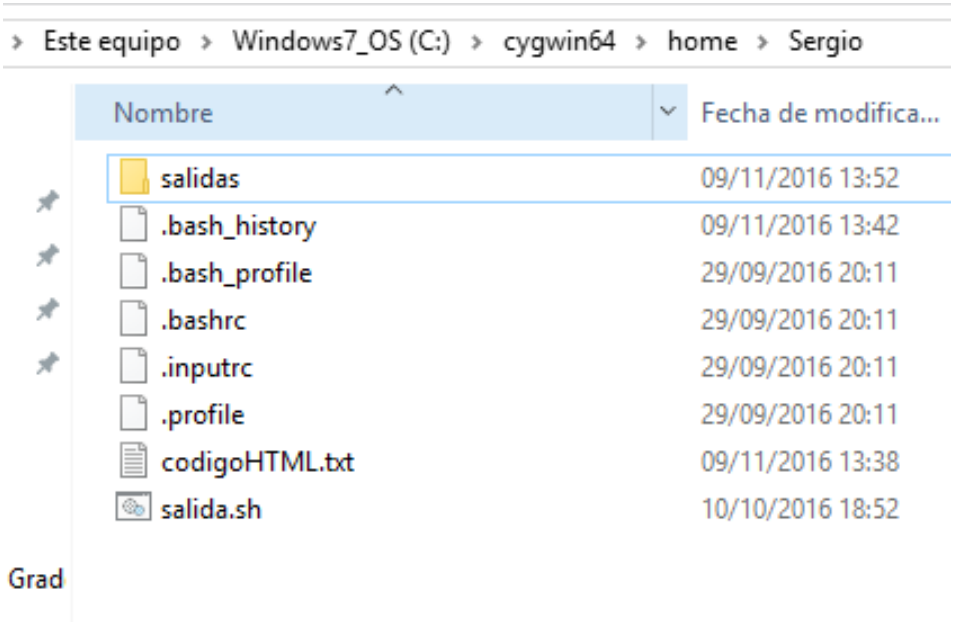

Ilustración 78: Distribución del directorio antes de ejecutar el script

Abrimos Cygwin64 y ejecutamos el script *salida.sh*.

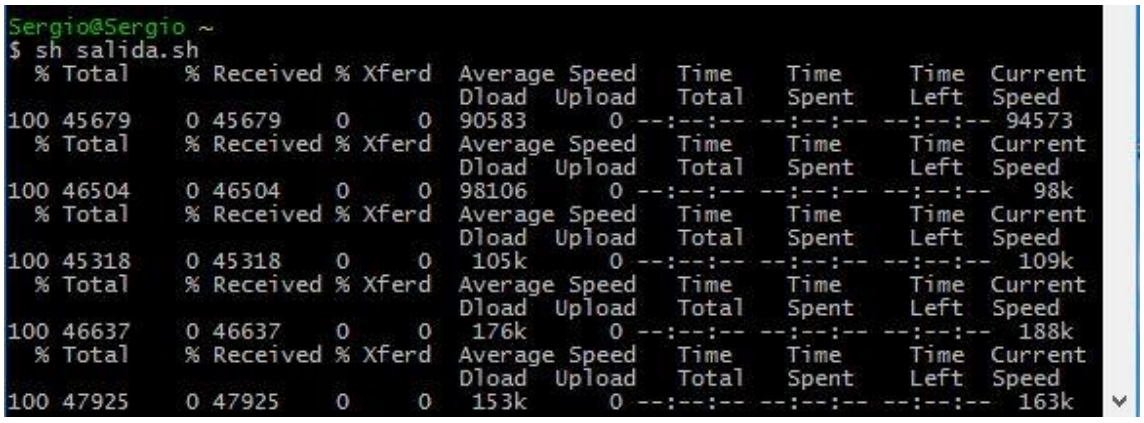

Ilustración 79: Ejecución del script

De este modo obtenemos, en la carpeta *salidas*, el código de todas las páginas de los miembros del Dpto. de Informática.

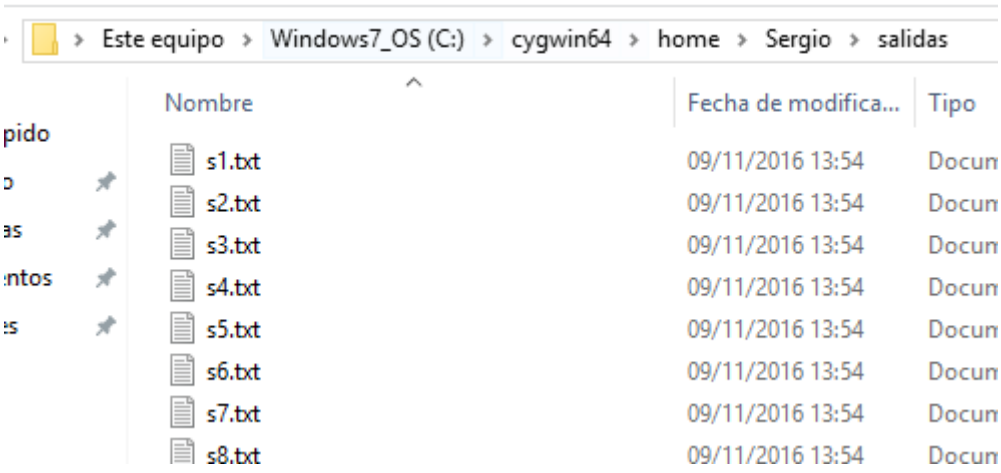

Ilustración 80: Distribución de la carpeta salidas

Sin embargo, al inspeccionar el código descargado de cada miembro del departamento, pude observar que, al contrario que el resto de los campos como el nombre, el despacho, el teléfono, etc., el email no venía en claro, al parecer para evitar que programas de spam descarguen el código, accedan al correo e inunden de información basura a los propietarios de los mismos. El correo venía codificado con HTML Codes, por lo que para decodificarlo era necesario usar la función de PHP *html_entity_decode*.

PASO 4

Para decodificar los correos, antes tenemos que almacenarlos. Por ello, ejecutamos *codigo2.c*. Este código recoge, de cada archivo almacenado en la carpeta *salidas*, una cadena de caracteres con el correo codificado, y lo vuelca en el archivo *codigoPHP.php*.

Ilustración 81: Ejecución de código2

PASO 5

Después de instalar un entorno para ejecutar PHP (ver *INSTALACIÓN DEL ENTORNO PHP*), accedemos con la consola de comandos a la carpeta donde se encuentre dicho entorno y ejecutamos *codigoPHP.php*, volcando los resultados en el archivo *correos.txt*.

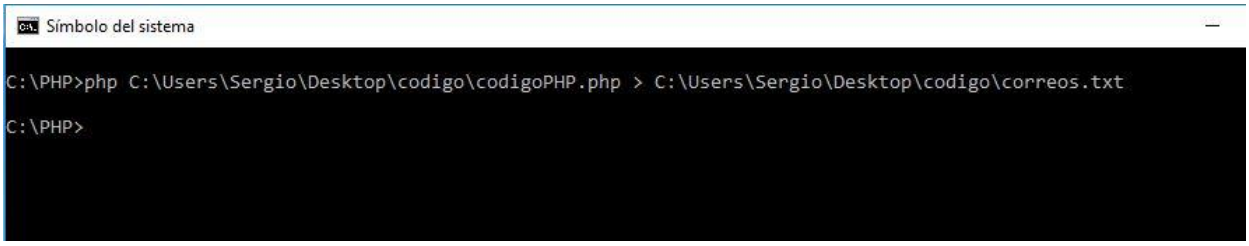

Ilustración 82: Ejecución del código PHP

PASO 6

Ejecutamos *código3.c*. Este código recoge, de cada archivo almacenado en la carpeta *salidas* y del archivo *correos.txt*, toda la información de cada miembro del Dpto. de Informática, y la vuelca sobre el archivo *datosLDAP.txt*.

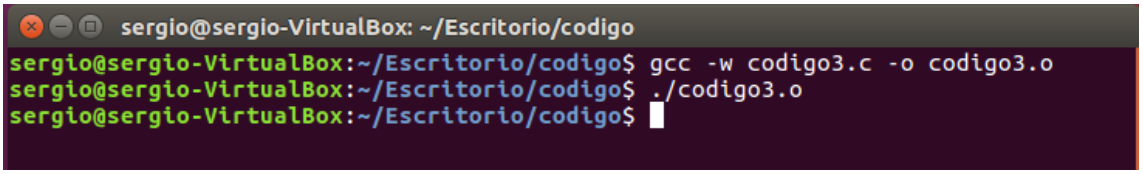

Ilustración 83: Ejecución de código3

Finalmente, es en el archivo *datosLDAP.txt* donde tenemos la información necesaria para realizar las pruebas en cada uno de los sistemas NoSQL elegidos con anterioridad.

Apéndice VI - Ejemplos prácticos de Cassandra

Inserción de datos desde un archivo

```
cqlsh:ldapspace> COPY ldap(nombre,apellidos,email,telefono,campus,edificio,despacho,categoria,area_de_conocimiento) FROM '/home/sergio/Escritorio/result.csv';
Using 1 child processes

Starting copy of ldapspace.ldap with columns [nombre, apellidos, email, telefono, campus, edificio, despacho, categoria, area_de_conocimiento].
Processed: 142 rows; Rate:     199 rows/s; Avg. rate:     309 rows/s
142 rows imported from 1 files in 0.459 seconds (0 skipped).
cqlsh:ldapspace>
```

Ilustración 84: Inserción de datos en Cassandra

Actualizar un usuario

```
cqlsh:ldapspace> select * from ldap where despacho='4.0.F11' ALLOW FILTERING;

 email                     | apellidos             | area_de_conocimiento                                                | campus   | categoria              | despacho | edificio       | nombre  | telefono
---------------------------+-----------------------+---------------------------------------------------------------------+----------+------------------------+----------+----------------+---------+-----------
     javier@lab.inf.uc3m.es |                  null |                                                                null | Leganés | Becario de laboratorio |  4.0.F11 | Torres Quevedo |  Javier | 916249061
       rfuentes@pa.uc3m.es |       Fuentes Astorga |                                                                null | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo | Roberto | 916249061
 oscar.perez.alonso@uc3m.es |          Pérez Alonso |                          Arquitectura y Tecnología de Computadores | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo |   Óscar | 916249061
      jorge@lab.inf.uc3m.es |       Guerrero Ramos | Ciencia de la Computación e Inteligencia Artificial | Leganés | Becario de laboratorio |  4.0.F11 | Torres Quevedo |   Jorge | 916249061
      jaime@lab.inf.uc3m.es | Pons Bailly-Bailliere |                                                                null | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo |   Jaime | 916249061

(5 rows)
cqlsh:ldapspace> update ldap set apellidos='Calvo Camuñas' where email='javier@lab.inf.uc3m.es';
cqlsh:ldapspace> select * from ldap where despacho='4.0.F11' ALLOW FILTERING;

 email                     | apellidos             | area_de_conocimiento                                                | campus   | categoria              | despacho | edificio       | nombre  | telefono
---------------------------+-----------------------+---------------------------------------------------------------------+----------+------------------------+----------+----------------+---------+-----------
     javier@lab.inf.uc3m.es |         Calvo Camuñas |                                                                null | Leganés | Becario de laboratorio |  4.0.F11 | Torres Quevedo |  Javier | 916249061
       rfuentes@pa.uc3m.es |       Fuentes Astorga |                                                                null | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo | Roberto | 916249061
 oscar.perez.alonso@uc3m.es |          Pérez Alonso |                          Arquitectura y Tecnología de Computadores | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo |   Óscar | 916249061
      jorge@lab.inf.uc3m.es |       Guerrero Ramos | Ciencia de la Computación e Inteligencia Artificial | Leganés | Becario de laboratorio |  4.0.F11 | Torres Quevedo |   Jorge | 916249061
      jaime@lab.inf.uc3m.es | Pons Bailly-Bailliere |                                                                null | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo |   Jaime | 916249061

(5 rows)
cqlsh:ldapspace>
```

Ilustración 85: Actualización de un usuario en Cassandra

Borrar un usuario

```
cqlsh:ldapspace> select * from ldap where despacho='4.0.F11' ALLOW FILTERING;

 email                     | apellidos             | area_de_conocimiento                                                | campus   | categoria              | despacho | edificio       | nombre  | telefono
---------------------------+-----------------------+---------------------------------------------------------------------+----------+------------------------+----------+----------------+---------+-----------
     javier@lab.inf.uc3m.es |         Calvo Camuñas |                                                                null | Leganés | Becario de laboratorio |  4.0.F11 | Torres Quevedo |  Javier | 916249061
       rfuentes@pa.uc3m.es |       Fuentes Astorga |                                                                null | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo | Roberto | 916249061
 oscar.perez.alonso@uc3m.es |          Pérez Alonso |                          Arquitectura y Tecnología de Computadores | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo |   Óscar | 916249061
      jorge@lab.inf.uc3m.es |       Guerrero Ramos | Ciencia de la Computación e Inteligencia Artificial | Leganés | Becario de laboratorio |  4.0.F11 | Torres Quevedo |   Jorge | 916249061
      jaime@lab.inf.uc3m.es | Pons Bailly-Bailliere |                                                                null | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo |   Jaime | 916249061

(5 rows)
cqlsh:ldapspace> delete from ldap  where email='javier@lab.inf.uc3m.es';
cqlsh:ldapspace> select * from ldap where despacho='4.0.F11' ALLOW FILTERING;

 email                     | apellidos             | area_de_conocimiento                                                | campus   | categoria              | despacho | edificio       | nombre  | telefono
---------------------------+-----------------------+---------------------------------------------------------------------+----------+------------------------+----------+----------------+---------+-----------
       rfuentes@pa.uc3m.es |       Fuentes Astorga |                                                                null | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo | Roberto | 916249061
 oscar.perez.alonso@uc3m.es |          Pérez Alonso |                          Arquitectura y Tecnología de Computadores | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo |   Óscar | 916249061
      jorge@lab.inf.uc3m.es |       Guerrero Ramos | Ciencia de la Computación e Inteligencia Artificial | Leganés | Becario de laboratorio |  4.0.F11 | Torres Quevedo |   Jorge | 916249061
      jaime@lab.inf.uc3m.es | Pons Bailly-Bailliere |                                                                null | Leganés | Técnico de laboratorio |  4.0.F11 | Torres Quevedo |   Jaime | 916249061

(4 rows)
cqlsh:ldapspace>
```

Ilustración 86: Borrado de un usuario en Cassandra

Borrar todos los datos

```
cqlsh:ldapspace> truncate ldap;
cqlsh:ldapspace> select * from ldap;

 email | apellidos | area_de_conocimiento | campus | categoria | despacho | edificio | nombre | telefono
-------+-----------+----------------------+--------+-----------+----------+----------+--------+----------

(0 rows)
cqlsh:ldapspace>
```

Ilustración 87: Borrado de los datos en Cassandra

Buscar usuarios

```
cqlsh:ldapspace> select * from ldap where nombre='Javier' ALLOW FILTERING;

 email                  | apellidos       | area_de_conocimiento                                | campus   | categoria                          | despacho | edificio       | nombre | telefono
------------------------+-----------------+-----------------------------------------------------+----------+------------------------------------+----------+----------------+--------+-----------
  javier@lab.inf.uc3m.es |    Calvo Camuñas |                                                null | Leganés |            Becario de laboratorio |  4.0.F11 | Torres Quevedo | Javier | 916249061
 javier.delicado@uc3m.es | Delicado Huelva |                                                null | Leganés |                         Secretaria |  2.1.B05 |       Sabatini | Javier | 916249049
    jahuerta@inf.uc3m.es |    Huertas Tato |                                                null |    null | Personal investigador en formación |     null |           null | Javier |     null
    jsrodrig@inf.uc3m.es |  Sanz Rodríguez | Ciencia de la Computación e Inteligencia Artificial | Leganés |                  Profesor Asociado |    21C03 |       Sabatini | Javier | 916248832
     jfmunoz@inf.uc3m.es | Fernández Muñoz |          Arquitectura y Tecnología de Computadores | Leganés |                    Profesor Titular | 2.2.B.17 |       Sabatini | Javier | 916249497
    jgarciag@inf.uc3m.es |   García Guzmán |                      Lenguajes y Sistemas Informáticos | Leganés |                    Profesor Titular | 2.1.C.11 |       Sabatini | Javier | 916249988
      jcarbo@inf.uc3m.es |   Carbó Rubiera | Ciencia de la Computación e Inteligencia Artificial | Leganés |                    Profesor Titular |  2.1.B21 |       Sabatini | Javier | 916249105

(7 rows)
cqlsh:ldapspace> select * from ldap where categoria='Catedrático' ALLOW FILTERING;

 email                 | apellidos          | area_de_conocimiento                                | campus      | categoria   | despacho | edificio          | nombre        | telefono
-----------------------+--------------------+-----------------------------------------------------+-------------+-------------+----------+-------------------+---------------+-----------
  dborrajo@ia.uc3m.es |     Borrajo Millán | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Catedrático | 2.1.B.09 |          Sabatini |        Daniel | 916249459
    molina@ia.uc3m.es |      Molina López | Ciencia de la Computación e Inteligencia Artificial | Colmenarejo | Catedrático | 1.2.B.28 | Miguel de Unamuno |    José Manuel | 918561314
  llorens@inf.uc3m.es |    Llorens Morillo | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Catedrático |  2.1.B07 |          Sabatini | Juan Bautista | 916249498
 jcarrete@inf.uc3m.es |    Carretero Pérez |          Arquitectura y Tecnología de Computadores |    Leganés | Catedrático |  2.2.A25 |          Sabatini |         Jesús | 916249458
      aedo@ia.uc3m.es |        Aedo Cuevas | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Catedrático | 2.2.A.19 |          Sabatini |       Ignacio | 916249490
      pdp@inf.uc3m.es |       Díaz Pérez | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Catedrático | 2.2.A.21 |          Sabatini |        Paloma | 916249456
   arturo@inf.uc3m.es | Ribagorda Garnacho | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Catedrático |  2.2.A23 |          Sabatini |        Arturo | 916249463
     isasi@ia.uc3m.es |      Isasi Viñuela | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Catedrático | 2.1.B.13 |          Sabatini |         Pedro | 916249455
 fgcarbal@inf.uc3m.es | García Carballeira |          Arquitectura y Tecnología de Computadores |    Leganés | Catedrático |  2.2.B19 |          Sabatini |         Félix | 916249060
  amescua@inf.uc3m.es |    de Amescua Seco | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Catedrático | 2.1.C.15 |          Sabatini |       Antonio | 916249461

(10 rows)
cqlsh:ldapspace>
```

Ilustración 88: Buscar usuarios por un atributo en Cassandra

```
cqlsh:ldapspace> select * from ldap;

 email                  | apellidos             | area_de_conocimiento                                | campus      | categoria                          | despacho | edificio          | nombre
------------------------+-----------------------+-----------------------------------------------------+-------------+------------------------------------+----------+-------------------+--------------
---
      antonsan@pa.uc3m.es |       Sánchez Santiago |                                                null |       null |                  Profesor Asociado |     null |              null |   Antonio Javi
er
      jllopez@inf.uc3m.es |         López Cuadrado |                      Lenguajes y Sistemas Informáticos |    Leganés |                 Profesor Visitante |    22A24 |          Sabatini |        José Lu
is
     clinares@inf.uc3m.es |          Linares López | Ciencia de la Computación e Inteligencia Artificial |    Leganés |                    Profesor Titular |  2.2.B09 |          Sabatini |           Carl
os
     abaldomi@inf.uc3m.es |       Baldominos Gómez | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Personal investigador en formación | 2.2.C01B |          Sabatini |        Alejand
ro
     jofuente@inf.uc3m.es |         Fuentez Cortez |                                                null |       null |                 Profesor Visitante |     null |              null |        José Re
ne
     gdugarte@inf.uc3m.es |           Dugarte Peña |                      Lenguajes y Sistemas Informáticos |    Leganés | Personal investigador en formación |  2.1.C02 |          Sabatini |     Germán Len
in
      igalvan@inf.uc3m.es |           Galván León | Ciencia de la Computación e Inteligencia Artificial |    Leganés |                    Profesor Titular | 2.2.B.25 |          Sabatini |       Inés Mar
ia
     gcarrasc@inf.uc3m.es | Carrascal de las Heras |                                                null | Colmenarejo |                  Profesor Asociado |  1.2.B17 | Miguel de Unamuno |            Gin
es
```

Ilustración 89: Buscar todos los usuarios en Cassandra

Mostrar resultados en la interfaz

```
cqlsh:ldapspace> select * from ldap;

 email                  | apellidos             | area_de_conocimiento                                | campus      | categoria                          | despacho | edificio          | nombre
------------------------+-----------------------+-----------------------------------------------------+-------------+------------------------------------+----------+-------------------+--------------
---
      antonsan@pa.uc3m.es |       Sánchez Santiago |                                                null |       null |                  Profesor Asociado |     null |              null |   Antonio Javi
er
      jllopez@inf.uc3m.es |         López Cuadrado |                      Lenguajes y Sistemas Informáticos |    Leganés |                 Profesor Visitante |    22A24 |          Sabatini |        José Lu
is
     clinares@inf.uc3m.es |          Linares López | Ciencia de la Computación e Inteligencia Artificial |    Leganés |                    Profesor Titular |  2.2.B09 |          Sabatini |           Carl
os
     abaldomi@inf.uc3m.es |       Baldominos Gómez | Ciencia de la Computación e Inteligencia Artificial |    Leganés | Personal investigador en formación | 2.2.C01B |          Sabatini |        Alejand
ro
     jofuente@inf.uc3m.es |         Fuentez Cortez |                                                null |       null |                 Profesor Visitante |     null |              null |        José Re
ne
     gdugarte@inf.uc3m.es |           Dugarte Peña |                      Lenguajes y Sistemas Informáticos |    Leganés | Personal investigador en formación |  2.1.C02 |          Sabatini |     Germán Len
in
      igalvan@inf.uc3m.es |           Galván León | Ciencia de la Computación e Inteligencia Artificial |    Leganés |                    Profesor Titular | 2.2.B.25 |          Sabatini |       Inés Mar
ia
     gcarrasc@inf.uc3m.es | Carrascal de las Heras |                                                null | Colmenarejo |                  Profesor Asociado |  1.2.B17 | Miguel de Unamuno |            Gin
es
```

Ilustración 90: Salida de resultados en Cassandra

Exportar datos a un archivo

```
cqlsh:ldapspace> COPY ldap(nombre,apellidos,email,telefono,campus,edificio,despacho,categoria,area_de_conocimiento) TO '/home/sergio/Escritorio/salida';
Using 1 child processes

Starting copy of ldapspace.ldap with columns [nombre, apellidos, email, telefono, campus, edificio, despacho, categoria, area_de_conocimiento].
Processed: 142 rows; Rate:      488 rows/s; Avg. rate:      138 rows/s
142 rows exported to 1 files in 1.042 seconds.
cqlsh:ldapspace>
```

Ilustración 91: Exportar datos a un archivo en Cassandra

Restringir acceso por credenciales y Gestionar permisos de usuarios

Para demostrar que Cassandra cumple con esta característica, hemos realizado un proceso de creación de roles y asignación de permisos. Estos comandos se encuentran mejor detallados en la documentación de Cassandra en [52].

Lo primero es modificar el archivo */etc/cassandra/cassandra.yaml*, para habilitar los procesos de autenticación y autorización. Para ello, cambiamos el parámetro *authenticator* de *AllowAllAuthenticator* a *PasswordAuthenticator*, y el parámetro *authorizer* de *AllowAllAuthorizer* a *CassandraAuthorizer*, tal y como se muestra en la siguiente imagen.

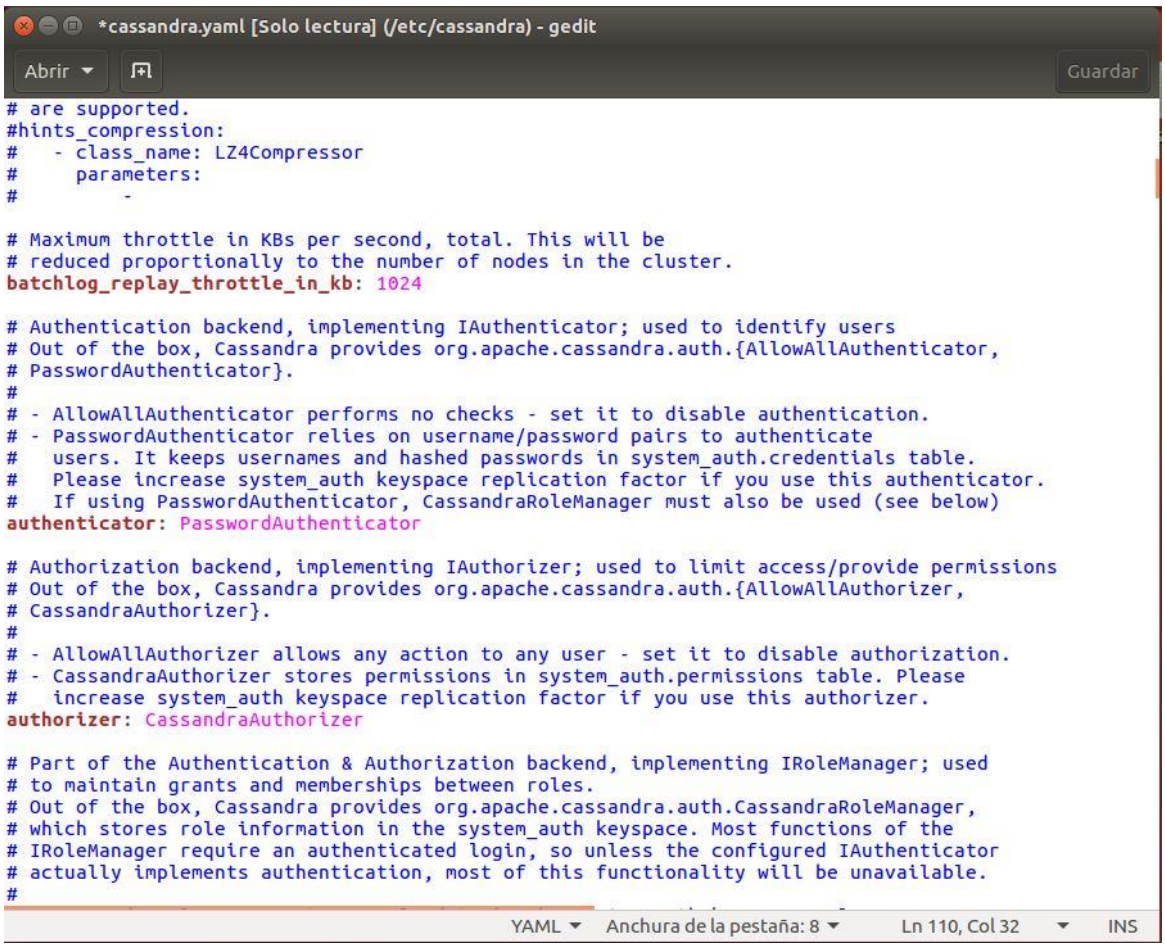

Ilustración 92: Código de *cassandra.yaml*

Ahora, si intentamos iniciar Cassandra con el comando *cqlsh* no nos dejará, puesto que debemos iniciar sesión. Como aún no tenemos usuarios creados, usaremos para entrar el superusuario por defecto, *cassandra*.

Ilustración 93: Inicio de sesión en Cassandra

Si listamos los roles, vemos que sólo tenemos el rol por defecto.

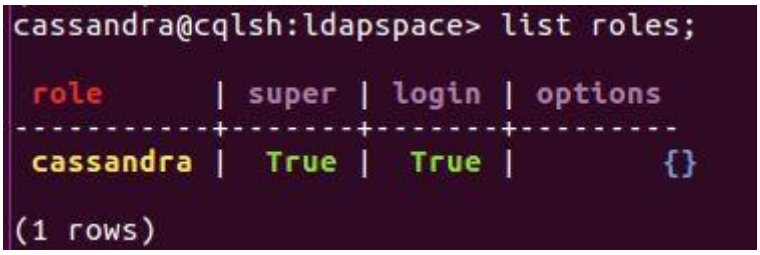

Ilustración 94: Roles por defecto

Creamos un nuevo rol *admin*. Al listar los roles, vemos que no tiene permisos de superusuario ni para iniciar sesión, mientras que el rol *cassandra* posee ambos.

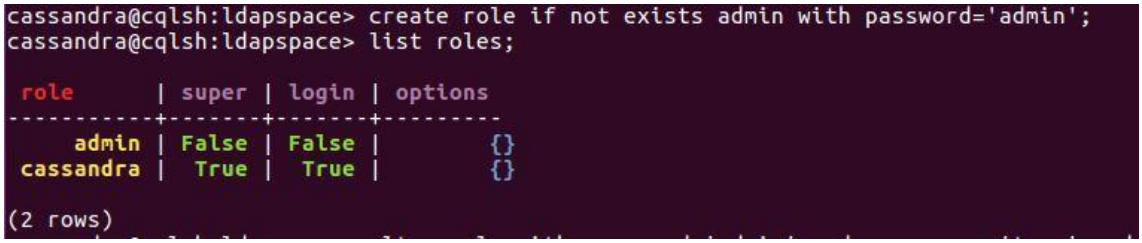

Ilustración 95: Crear rol *admin*

Le cambiamos los permisos al rol *admin* y le permitimos que inicie sesión y que sea superusuario.

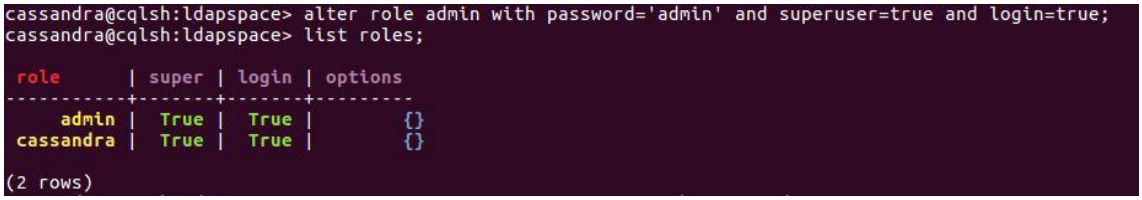

Ilustración 96: Asignar permisos al rol *admin*

Ahora iniciamos sesión con el nuevo rol para comprobar que, efectivamente, posee este permiso. Como ya tenemos el usuario administrador, borramos el superusuario por defecto, *cassandra*, para evitar problemas de seguridad.

```
sergio@sergio-VirtualBox:/etc/cassandra$ cqlsh -u admin -p admin
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.7 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
admin@cqlsh> drop role if exists cassandra;
admin@cqlsh> list roles;

 role  | super | login | options
-------+-------+-------+---------
 admin |  True |  True |       {}

(1 rows)
admin@cqlsh>
```

Ilustración 97: Iniciar sesión con el rol *admin*

Ahora vamos a crear el rol *user*, que tendrá sólo acceso de lectura a la base de datos. Si lo creamos sin asignarle permisos de login no podrá acceder a Cassandra.

```
admin@cqlsh> create role if not exists user with password='user';
admin@cqlsh> exit
sergio@sergio-VirtualBox:/etc/cassandra$ cqlsh -u user -p user
Connection error: ('Unable to connect to any servers', {'127.0.0.1': Authent
icationFailed(u'Failed to authenticate to 127.0.0.1: code=0100 [Bad credenti
als] message="user is not permitted to log in"',)})
sergio@sergio-VirtualBox:/etc/cassandra$
```

Ilustración 98: Crear nuevo rol *user*

Por tanto, tenemos que asignarle al rol *user* permisos de login. Una vez hecho, entramos con el nuevo rol.

```
sergio@sergio-VirtualBox:/etc/cassandra$ cqlsh -u admin -p admin
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.7 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
admin@cqlsh> alter role user with password='user' and login=true;
admin@cqlsh> list roles;

 role  | super | login | options
-------+-------+-------+---------
 admin |  True |  True |       {}
  user | False |  True |       {}

(2 rows)
admin@cqlsh> exit
sergio@sergio-VirtualBox:/etc/cassandra$ cqlsh -u user -p user
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.7 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
user@cqlsh>
```

Ilustración 99: Asignar permisos de login al rol *user*

Sin embargo, si intentamos leer o escribir en la base de datos, no nos deja, porque aún no tenemos los permisos necesarios.

```
user@cqlsh> use ldapspace;
user@cqlsh:ldapspace> select * from personas;
Unauthorized: code=2100 [Unauthorized] message="User user has no SELECT permission on <table l
dapspace.personas> or any of its parents"
user@cqlsh:ldapspace> insert into personas(nombre,apellidos,dni) values('a','b','2');
Unauthorized: code=2100 [Unauthorized] message="User user has no MODIFY permission on <table l
dapspace.personas> or any of its parents"
user@cqlsh:ldapspace> truncate personas;
Unauthorized: code=2100 [Unauthorized] message="User user has no MODIFY permission on <table l
dapspace.personas> or any of its parents"
user@cqlsh:ldapspace>
```

Ilustración 100: Comprobar permisos de acceso a la base de datos del rol *user*

Para hacerlo, volvemos a entrar como *admin* y le damos permisos para que ejecute la operación SELECT. Ahora, al entrar como *user* y hacer una búsqueda con el comando SELECT, sí podremos ver la información de la base de datos.

```
sergio@sergio-VirtualBox:/etc/cassandra$ cqlsh -u admin -p admin
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.7 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
admin@cqlsh> grant select on keyspace ldapspace to user;
admin@cqlsh> exit
sergio@sergio-VirtualBox:/etc/cassandra$ cqlsh -u user -p user
Connected to Test Cluster at 127.0.0.1:9042.
[cqlsh 5.0.1 | Cassandra 3.7 | CQL spec 3.4.2 | Native protocol v4]
Use HELP for help.
user@cqlsh> use ldapspace;
user@cqlsh:ldapspace> select * from personas;

 dni | apellidos | nombre
-----+-----------+--------
   1 |         b |      a

(1 rows)
user@cqlsh:ldapspace> insert into personas(nombre,apellidos,dni) values('a','b','2');
Unauthorized: code=2100 [Unauthorized] message="User user has no MODIFY permission on <table l
dapspace.personas> or any of its parents"
user@cqlsh:ldapspace> truncate personas;
Unauthorized: code=2100 [Unauthorized] message="User user has no MODIFY permission on <table l
dapspace.personas> or any of its parents"
user@cqlsh:ldapspace>
```

Ilustración 101: Asignar permiso de lectura al rol *user*

Por último, me gustaría hacer una aclaración. Como bien indican en la documentación de Cassandra [52], para versiones superiores a la 2.2 (en mi caso uso la 3.7) la autenticación y autorización está basada en roles, en vez de usuarios, por lo que cuando se hable de inicio de sesión con credenciales, serán las credenciales del rol asociado, ya que no se va a generar ningún usuario para este proyecto.

Bibliografía

[1] C. Requena, «NoSQL.es,» 1 Abril 2010. [En línea]. Available: http://www.nosql.es/blog/nosql/que-es-nosql.html. [Último acceso: Octubre 2016].

[2] C. Paramio, «GENBETA:dev,» 26 Abril 2011. [En línea]. Available: http://www.genbetadev.com/bases-de-datos/el-concepto-nosql-o-como-almacenar-tus-datos-en-una-base-de-datos-no-relacional. [Último acceso: Septiembre 2016].

[3] P. Sarathi, «Big Data - Made Simple,» 21 Julio 2014. [En línea]. Available: http://bigdata-madesimple.com/a-deep-dive-into-nosql-a-complete-list-of-nosql-databases/. [Último acceso: Octubre 2016].

[4] W. D. Sepúlveda, «Bases de datos NOSQL,» 27 Mayo 2013. [En línea]. Available: http://basesdedatosnosql.blogspot.com.es/. [Último acceso: Octubre 2016].

[5] S. Pérez, «Ondho,» 30 Marzo 2015. [En línea]. Available: https://www.ondho.com/claves-para-elegir-tu-base-de-datos-nosql/. [Último acceso: Octubre 2016].

[6] Rubenfa, «GENBETA:dev,» 27 Enero 2014. [En línea]. Available: http://www.genbetadev.com/bases-de-datos/bases-de-datos-nosql-elige-la-opcion-que-mejor-se-adapte-a-tus-necesidades. [Último acceso: Octubre 2016].

[7] N. T. (. d. «Neo4j,» [En línea]. Available: https://neo4j.com/developer/graph-database/. [Último acceso: Noviembre 2016].

[8] N. T. (. «Neo4j,» [En línea]. Available: https://neo4j.com/blog/aggregate-stores-tour/. [Último acceso: Noviembre 2016].

[9] M. Rouse, «TechTarget,» Enero 2015. [En línea]. Available: http://searchdatacenter.techtarget.com/es/definicion/Base-de-datos-relacional. [Último acceso: Octubre 2016].

[10] S. FullContact, «FullContact,» [En línea]. Available: https://support.fullcontact.com/knowledgebase/articles/332361-what-are-google-s-contact-storage-limits. [Último acceso: Octubre 2016].

[11] S. Office, «Microsoft Office,» [En línea]. Available: https://support.office.com/es-es/article/Especificaciones-y-l%C3%ADmites-de-Excel-ca36e2dc-1f09-4620-b726-67c00b05040f#bmworksheetworkbook. [Último acceso: Noviembre 2016].

[12] Ferdy, «El diario de Ferdy,» 25 Febrero 2011. [En línea]. Available: http://www.rodenas.org/ferdyblog/2011/02/25/el-teorema-de-cap/. [Último acceso: Octubre 2016].

[13] Rubenfa, «GENBETA:dev,» 28 Enero 2014. [En línea]. Available: http://www.genbetadev.com/bases-de-datos/nosql-clasificacion-de-las-bases-de-datos-segun-el-teorema-cap. [Último acceso: Octubre 2016].

[14] N. T. (. «Neo4j,» [En línea]. Available: https://neo4j.com/customers/. [Último acceso: Noviembre 2016].

[15] C. C. «Cenatic,» [En línea]. Available: http://catalogo.cenatic.es/index.php?option=com_zoo&task=item&item_id=245&Itemid=172. [Último acceso: Noviembre 2016].

[16] R. Martínez, «Errores que todos cometemos en un proyecto Big Data con Spark y cómo arreglarlos,» de *Apache Spark Meetup*, Madrid, 2016.

[17] JoSek, «Josek,» [En línea]. Available: http://www.josek.net/2010/03/cassandra-nsql/. [Último acceso: Noviembre 2016].

[18] Objectivity, «Objectivity,» [En línea]. Available: http://www.objectivity.com/partners/. [Último acceso: Noviembre 2016].

[19] M. (. «MongoDB,» [En línea]. Available: https://www.mongodb.com/who-uses-mongodb. [Último acceso: Octubre 2016].

[20] Apache HBase, «HBase,» [En línea]. Available: https://hbase.apache.org/poweredbyhbase.html. [Último acceso: Octubre 2016].

[21] Redis, «Redis,» [En línea]. Available: http://redis.io/topics/whos-using-redis. [Último acceso: Septiembre 2016].

[22] European Space Agency, «esa,» [En línea]. Available: http://www.esa.int/ESA. [Último acceso: Diciembre 2016].

[23] Gobierno de España, «Portal de Administración Electrónica,» [En línea]. Available: https://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Metrica_v3.html. [Último acceso: Diciembre 2016].

[24] ISO, «ISO,» [En línea]. Available: http://www.iso.org/iso/catalogue_detail?csnumber=43447. [Último acceso: Diciembre 2016].

[25] AENOR, «Aenor,» [En línea]. Available: http://www.aenor.es/AENOR/certificacion/calidad/calidad_software_15504.asp. [Último acceso: Diciembre 2016].

[26] JGraph Ltd., «Draw.io,» [En línea]. Available: https://www.draw.io/. [Último acceso: Diciembre 2016].

[27] Gobierno de España, «Agencia Estatal Boletín Oficial del Estado,» [En línea]. Available: https://www.boe.es/. [Último acceso: Enero 2017].

[28] Apache Cassandra, «Cassandra,» [En línea]. Available: http://cassandra.apache.org/download/. [Último acceso: Diciembre 2016].

[29] Oposiciones TIC, «Oposiciones TIC,» [En línea]. Available: https://oposicionestic.blogspot.com.es/2012/08/en-1975-el-organismo-de-estandarizacion.html. [Último acceso: Diciembre 2016].

[30] M. Zaforas, «Paradigma Digital,» 17 Marzo 2016. [En línea]. Available: https://www.paradigmadigital.com/dev/cassandra-la-dama-de-las-bases-de-datos-nosql/. [Último acceso: Enero 2017].

[31] J. Mellado, «Inmensia,» 27 Marzo 2010. [En línea]. Available: http://inmensia.com/blog/20100327/desmitificando_a_cassandra.html. [Último acceso: Enero 2017].

[32] Google Sites, «Uegoman,» [En línea]. Available: https://sites.google.com/site/uegoman/modelo-de-datos-de-cassandra. [Último acceso: Enero 2017].

[33] S. Aragonés, «GitHub,» 11 Noviembre 2016. [En línea]. Available: https://github.com/sergioat7/RepoTFG. [Último acceso: Noviembre 2016].

[34] A. Rajagopalan, «Getting Started With Cassandra,» 26 Junio 2011. [En línea]. Available: http://gettingstartedwithcassandra.blogspot.com.es/2011/06/cassandra-keyspaces-what-are-they.html. [Último acceso: Diciembre 2016].

[35] DataStax, Inc., «Datastax (CQL reference),» 28 Noviembre 2016. [En línea]. Available: https://docs.datastax.com/en/cql/3.1/cql/cql_reference/cqlReferenceTOC.html. [Último acceso: Diciembre 2016].

[36] V. Omanashvili, «JSON GENERATOR,» [En línea]. Available: http://www.json-generator.com/. [Último acceso: Noviembre 2016].

[37] E. Mill, «Konklone,» [En línea]. Available: https://konklone.io/json/. [Último acceso: Noviembre 2016].

[38] V. J. Fernández, «Aventuras y desventuras de un informático despistado,» 17 Noviembre 2012. [En línea]. Available: https://vjavierf.wordpress.com/2012/11/17/virtualbox-acceder-por-rdp-a-una-

vm/. [Último acceso: Enero 2017].

[39] DataStax, Inc, «Datastax Academy,» [En línea]. Available: https://academy.datastax.com/planet-cassandra//cassandra. [Último acceso: Enero 2017].

[40] Gobierno de España, «Seguridad Social,» [En línea]. Available: http://www.seg-social.es/Internet_1/Trabajadores/CotizacionRecaudaci10777/Basesytiposdecotiza36537/index.htm. [Último acceso: Enero 2017].

[41] L. «La Web del Programador,» [En línea]. Available: http://www.lawebdelprogramador.com/diccionario/. [Último acceso: Noviembre 2016].

[42] L. Alegsa, «Alegsa,» 27 Noviembre 2008. [En línea]. Available: http://www.alegsa.com.ar/Diccionario/diccionario.php. [Último acceso: Noviembre 2016].

[43] «Oracle,» [En línea]. Available: http://www.oracle.com/webfolder/technetwork/tutorials/obe/cloud/objectstorage/installing_cURL/installing_cURL_on_Cygwin_on_Windows.html#section1. [Último acceso: Septiembre 2016].

[44] PHP, «PHP,» [En línea]. Available: http://windows.php.net/download/. [Último acceso: Noviembre 2016].

[45] Oracle, «VirtualBox,» [En línea]. Available: https://www.virtualbox.org/wiki/Downloads. [Último acceso: Septiembre 2016].

[46] Ubuntu, «Ubuntu,» [En línea]. Available: http://www.ubuntu.com/download. [Último acceso: Octubre 2016].

[47] T. Fox, «DigitalOcean,» 21 Octubre 2015. [En línea]. Available: https://www.digitalocean.com/community/tutorials/how-to-install-cassandra-and-run-a-single-node-cluster-on-ubuntu-14-04. [Último acceso: Octubre 2016].

[48] R. C. Mora, «AdictosAlTrabajo.com,» 5 Diciembre 2013. [En línea]. Available: https://www.adictosaltrabajo.com/tutoriales/primeros-pasos-apache-cassandra/. [Último acceso: Octubre 2016].

[49] «Departamento de Informática UC3M,» [En línea]. Available: http://www.inf.uc3m.es/personal. [Último acceso: Octubre 2016].

[50] «HTML Codes Table,» [En línea]. Available: http://www.ascii.cl/htmlcodes.htm. [Último acceso: Diciembre 2016].

[51] «Curl,» [En línea]. Available: https://curl.haxx.se/docs/httpscripting.html. [Último

acceso: Octubre 2016].

[52] DataStax, Inc, «Datastax (Securing a table),» 29 Diciembre 2016. [En línea]. Available: http://docs.datastax.com/en/cql/3.3/cql/cql_using/useSecureTOC.html. [Último acceso: Enero 2017].